המכללה האקדמית הדסה ירושלים ע.ר
Hadassah Academic College Jerusalem R.A

HAC
אקדמיה מובילה בעולם משתנה

Department of Computer Science

B.Sc. and M.Sc. Programs for the academic year 5780 (2020-2021)

Jerusalem, Elul 5779, August 2019

www.hac.ac.il

רח' הנביאים 37, ת.ד. 1114, ירושלים 9101001 02-6291911 02-6250619
37 Hanevi'im Street, P.O. Box 1114, Jerusalem 9101001 +972-2-6291911 +972-2-6250619

Table of Contents

I. Introduction

The Department of Computer Science at the Hadassah Academic College offers study programs towards two degrees: (a) A Bachelor's Degree in Computer Science (B.Sc.) and (b) a Master's Degree in Computer Science (M.Sc.).

The department initially opened its doors in 1995 and its first class graduated in 1998. Until 2001, the department granted a BA degree in Computer Science, and in 2002 began offering a BSc degree. In 2008 the department opened its MSc degree program in Computer Science. The department has trained hundreds of graduates in the field of Computer Science who have successfully entered the work force in Jerusalem and throughout Israel. Many graduates continue their studies towards advanced degrees at distinguished universities in Israel and abroad, and others complete their MSc degree at the Hadassah Academic College.

Both the BSc and MSc degree programs in the Department of Computer Science include studies in software and programming, hardware, systems, mathematics, theoretical computer science, elective studies in advanced areas of computerization, final projects, and general topics. Our students consequently benefit from a combination of several worlds – a solid foundation in mathematics and theoretical computer science, thorough and immersive knowledge of hardware and software in the world of modern computers, and a variety of academic enrichment courses.

The department places great importance on cultivating personal relations with students, so that lecturers and students become well-acquainted during the course of study. Many hours are devoted to problem solving sessions and workshops alongside formal classroom studies. Lecturers and teaching assistants make themselves available in the computer laboratories outside classroom hours, enabling students to benefit directly and extensively from their knowledge. The department's computer laboratories are regularly renovated and made available to students at all time.

Studies in the Department of Computer Science continue for three years (six semesters) and include 140 credits. Students who work can study on a part-time basis and extend their studies beyond three years. Students who have already earned a two-year diploma in practical software engineering and meet the entry requirements for the BSc program are eligible for an exemption of up to 20 credits.

The department also offers a study program towards an MSc in Computer Science. This program is designed for candidates who hold a BSc in Computer Science or other allied fields of science or engineering. The program is designed to enable students who work in the field of computers to combine work and study. The MSc program continues for two to four years (four to eight semesters) and includes 48 credits.

Useful College Telephone Numbers

College President	Prof. Bertold Friedlander	(02) 629-1975
Vice President & Academic Director	Dr. Tzachi Milgrom	(02) 629-1978
Head of School of Computer Science	Prof. Michel Bercovier	
Computer Science Department Chair	Prof. Michael Berman	(02) 629-1953
Head of MSc Program	Dr. Solange Karsenty	Via Department Coordinator
Department Coordinator	Ravit Dor	(02) 629-1931
Dean of Students	Dr. Simcha Rozen	(02) 629-1307
Student Advisor	Tzofit Chaim	(02) 629-1306
Director of Challenge Center	Ofra Rotem	(02) 629-1938
Director of Student Administration	Yael Catalan	(02) 629-1964
Student Administration Coordinator	Eynav Rosenblum	(02) 629-1944
Chief Financial Officer	Maya Shraga Albalak	(02) 629-1993
Tuition Department	Asaf Malkosh	(02) 629-1990
Head of English Department	Nourit Melcer-Padon	(02) 629-1310
Director of Information Resource Center	Eric Royi	(02) 629-1303
Deputy Director Logistics & Information	Gad Singer	(02) 629-1970
Information Office		(02) 629-1911

Academic Calendar 5780 (2020-2021) – Hadassah College (HaNevi'im Campus)

See (in Hebrew): https://tinyurl.com/y5qv373q/

II. Department Curriculum

General description of Department Programs

Curriculum for BSc in Computer Science

The curriculum for the BSc in Computer Science is intended for those with a Bagrut matriculation certificate (or an equivalent). The planned duration of this course is three years (six semesters). This track includes a final project. Scope of study in this track is 140 credits.

Students with a two-year 'Practical Software Engineering' degree who meet the admission requirements of the 'Engineer Completion' program will enjoy an exemption of 20 credits out of the above.

BSc in Computer Science: Fields of study

- **Software & Programming:** Basic and advanced studies distributed throughout the degree program. These are mandatory and are required to the same extent across all curricula.
- **Hardware & Systems:** Basic studies spread out over the degree program. Both the regular and Haredi tracks include several required courses in hardware and systems, with exemptions in the completion track.
- **Mathematics:** Fundamental courses in mathematics are taught in the first and second year. These studies are mandatory and are required to the same extent across all curricula.
- **Theoretical Computer Science:** Elementary and advanced studies distributed throughout the program. These are mandatory and all are required in all curriculum tracks.
- **Elective Course in Computer Sciences:** Advanced studies are taken in the third year.
- **Projects:** A final project is carried out in the final year of the study program.
- **General Topics:** As part of the studies for a degree in Computer Science, students are required to complete a number of elective courses on general topics. These courses can be spread out over the entire degree program.

Curriculum for MSc in Computer Sciences

MSc Degree: The curriculum for the MSc in Computer Science is intended for those with a BSc in Computer Science or a related scientific or engineering field. The MSc Computer Science curriculum is a program of between two years (four semesters) to four years (eight semesters). The program has been constructed in a manner that will enable people who work in the field of computers to combine their studies and their work. It includes mandatory courses, elective courses, a seminar, final project and a comprehensive final exam at the end of studies.

General comments regarding all curricula

A. Exemptions from certain courses or credits may be obtained based on prior studies. Procedures and guidelines for exemption from courses or credits are provided in the College Regulations.

B. Curricula in the department are dynamic and updated year by year. The Computer Science Teaching Committee may revise and update curricula as required – syllabuses, scope of study and the curriculum for each year.

Degree requirements in the department by topic

BSc (three years):

TOPIC	CREDITS
Mathematics	32
Theoretical Computer Science	25
Software & Programming	32
Hardware & Systems	25
Computer Science Elective Course & Seminar	8
Final Project	8
Total Computer Science	**130**
General Topics	10
Total	**140**

MSc (two years):

TOPICS	CREDITS
Theoretical Computer Science – mandatory	6
Software & Programming – mandatory	6
Hardware & Systems – mandatory	6
Computer Science Elective Course	20
Final Project	10
Total Computer Sciences	**48**

BSc Program in the Department

First Year

SEMESTER A			SEMESTER B		
COURSE	CREDITS	WEEKLY HOURS	COURSE	CREDITS	WEEKLY HOURS
Mathematical Tools for Computer Science	4	2 + 3	Calculus: Single Variable Calculus	4	2 + 3
Linear Algebra A	4	2 + 3	Linear Algebra B	4	2 + 3
Introduction to Computer Science	4	2 + 3	Introduction to Theoretical Computer Science	4	2 + 3
Discrete Mathematics	5	3+2+4	Modular Programming	5	3+2+4
Digital Systems	4	4	Hardware/Software Systems & Assembly Language Programming	4	4
Mathematics Workshop A	0	2	Mathematics Workshop B	0	2
English		0 to 6	English		0 to 6
Total	**21**	**30**	**Total**	**21**	**30**

Second Year

SEMESTER A			SEMESTER B		
COURSE	CREDITS	WEEKLY HOURS	COURSE	CREDITS	WEEKLY HOURS
Calculus: Applications of Integrals and Approximations	4	2 + 3	Calculus: Curves & Surfaces	4	2 + 3
Algebraic Structures	4	4	Probability Theory	4	4
Data Structures	4	2 + 3	Algorithms	4	2 + 3
Introduction to Object Oriented Programming & Software Engineering	5	2+2+4	Object Oriented Programming & Game Development	5	2+2+4
System Programming & Introduction to Parallel Programming	5	2+2+4	Scripting Operating Systems & Programming	5	2+2+4
Scientific Writing & Presentation	2	2	General Elective Course A	2	2
English		0 to 4	English		0 to 4
Total	**24**	**32**	**Total**	**24**	**32**

Third Year					
SEMESTER A			**SEMESTER B**		
COURSE	**CREDITS**	**WEEKLY HOURS**	**COURSE**	**CREDITS**	**WEEKLY HOURS**
Automata and Formal Languages	3	3	Computability & Computational Complexity	3	3
Logic for Computer Science	3	3	Databases	4	4
Internet Programming A	4	4	Internet Programming B	4	4
Computer Networking	4	4	Computer Architecture	3	3
Computer Science Elective Course A	3	3	Computer Science Elective Course B	3	3
Final Project **(year course)**	8	3	Final Project **(continued)**		3
Computer Science Seminar	2	2	General Elective Course C	2	2
General Elective Course B	2	2	General Elective Course D	2	2
Total	**29**	**24**	**Total**	**19**	**24**

A. The courses ‘Scientific Writing & Presentation’ and ‘Computer Science Seminar’ may be taught in either semester A or semester B.
B. Each student must earn 8 credits in general elective courses other than the ‘Scientific Writing & Presentation’ course. Accumulation does not have to be by four courses of 2 credits each.
C. Two summer semesters will be held on the Strauss Campus. Each year, two courses will be taken during the summer semester (and not in semester A or B).

Students with a Practical Software Engineering Degree

Students on the BSc completion track follow the same curriculum as those on the regular track, with the exception of the following courses from which they are exempt:

A. Digital Systems (4 credits)
B. Hardware/Software Systems & Assembly Language Programming (4 credits)
C. System Programming & Introduction to Parallel Programming (5 credits)
D. General Elective Course (2 credits)
E. Students who achieve a grade of at least 75 in each of the courses ‘Introduction to Computer Science’ and ‘Modular Programming’ will be exempt from the course ‘Object Oriented Programming & Game Development’ (5 credits). Students who are not exempt from this course will instead be exempt from one elective course in computer sciences (3 credits) and one general elective course (2 credits).

MSc Program in the Department

First Year or Second Year					
SEMESTER A			**SEMESTER B**		
COURSE	**CREDITS**	**WEEKLY HOURS**	**COURSE**	**CREDITS**	**WEEKLY HOURS**
Computational Complexity	3	3	Object Oriented Analysis and Design	3	3
Computer Science Elective Course A	3	3	Protocols and Computer Networks	3	3
Computer Science Elective Course B	3	3	Computer Science Elective Course C	3	3
			Computer Science Elective Course D	3	3
Total	**9**	**9**	**Total**	**12**	**12**

Second Year or First Year					
SEMESTER A			**SEMESTER B**		
COURSE	**CREDITS**	**WEEKLY HOURS**	**COURSE**	**CREDITS**	**WEEKLY HOURS**
Advanced Algorithms	3	3	Advanced Computer Architecture	3	3
Software Engineering	3	3	Computer Science Elective Course F	3	3
Computer Science Seminar	2	2	Final Project	10	10
Computer Science Elective Course E	3	3			
Total	**11**	**11**	**Total**	**16**	**16**

Comments:

- The MSc curriculum includes six mandatory courses – Two in the theory of computer science: 'Computational Complexity' and 'Advanced Algorithms'; Two in the field of software: 'Software Engineering' and 'Object Oriented Analysis and Design'; and two in the field of systems: 'Advanced Computer Architecture' and 'Protocols and Computer Networks'.
- Courses in the MSc program can be completed over two to four years.
- The final project shall be completed under academic and scientific supervision of department faculty members.
- Upon completion of studies, a comprehensive examination is conducted that covers all areas of study in the curriculum.

III. Academic Regulations

The regulations listed below are based on Hadassah College's Academic Policies and Computer Science Department Regulations.

The regulations outlined below may be changed or revised at the discretion of the college and/or department.

All regulations are written in the male gender but refer to male and female students alike.

The department's teaching committee may, at its discretion, diverge from the procedures below.

Academic Department

An academic department is a framework that holds academic studies for a bachelor's degree and/or master's degree.

Computer Science Programs

The Computer Science Department offers the following study tracks:

- **Regular track** – three-year BSc studies for applicants with a matriculation certificate or an equivalent (students with a two-year 'Software Engineer' degree shall study on this track and be exempt from a number of credits).
- **MSc track** – two-year MSc track for students with a BSc in Computer Sciences or a related scientific or engineering field.

Admission to the Computer Science Department

Candidates who have met the admission conditions for one of the courses in the Computer Science Department will be accepted into a full-time program with the status of full-time student in this course.

Candidates who have not met conditions for admission to the department may, in exceptional cases, be admitted to a limited program with a temporary status of irregular student. This status requires the approval of the department's admissions committee and is valid for one academic year only. Students' curriculum for this academic year will be determined by the head of the department. At the end of the academic year, the student's status will be discussed by the department's pedagogical monitoring committee. Depending on the students' achievements, it is decided whether to transfer them to a full-time student status or to stop their studies in the department.

Study Duration

Studies in the Computer Science Department are conducted over the winter semester and spring semester. The Strauss Campus (Haredi Program) also holds a summer semester. Exams are conducted and projects are submitted at the end of each semester. Students must study the various subjects indicated in the program published by the department.

Attendance

The Computer Science Department requires a minimum of 80% attendance in the following courses:

- Mandatory English studies
- Scientific skill courses
- Computer science seminars

Lecturers in each course may determine attendance requirements in the course syllabus, at their discretion.

Students repeating courses due to a failing grade are required to attend 80% of classes, unless the course syllabus explicitly exempts them. It is the student's responsibility to obtain the lecturer's signature on an attendance form at the end of each class, and hand the signed form to the lecturer at the end of the course. A student who fails to do so will fail the course.

A student who is absent beyond the allowable hour limit will be considered as not meeting course requirements, and this without further additional notice.

Absence from courses with mandatory attendance may result in courses being invalidated.

Lecturers are entitled not to admit a late student.

Course Weight

Each course is given an academic weight expressed in credits. The weight of a course is the same for all students taking the course in a given academic year. The weight of a course may vary from one academic year to another. Credit allocations are made by the department's teaching committee. As a rule, the number of credits for a course is determined by the number of lecture and problem session hours on the course – one weekly semester hour in a lecture awards one credit and one weekly semester hour of problem session awards half a credit – however, there may be exceptions.

Study Programs

Each study program in the Computer Science Department is assigned a curriculum which includes mandatory courses and elective courses. Each course belongs to a field of study (mathematics, theoretical computer sciences, software and programming, hardware and systems, computer science elective courses etc.) and to a particular academic year within the program. Programs may change at the teaching committee's decision. Revised programs are published in the department newsletter at the start of each year.

Regular students must complete the courses in accordance with the curriculum on their study track. Students failing to comply with the curriculum in their study track shall receive a personalized curriculum.

Study Form

Registration for courses is through an online study form. The study form is a statement from students about the courses they intend to study during the year. This statement is binding for students with regards to academic and tuition requirements.

At the start of each academic year, all students will enter their curriculum into the information system. The curriculum will be forwarded to approval of an academic advisor. Where necessary, and under the academic advisor's guidance, the student will have to amend their registration. Study programs must comply with the following conditions:

- Courses in the are part of the curriculum in the track that students are studying.
- Students meet prerequisites for all courses included in their curriculum.
- The courses meet minimum requirements determined by the academic advisor.
- In the event the study form includes courses from different study years of students' curriculum – the academic year to which the courses belong is no more than one year ahead.
- Students whose curriculum has been dictated by the Department Head or Pedagogical Monitoring Committee will submit a study form detailing the prescribed program.

Any changes to the curriculum require authorization from the academic advisor. Requests to add or remove courses must be submitted to the academic advisor in accordance with the timetable prescribed in the student regulations.

Dropping a Course

Students will be defined as having dropped a course in the event that they begin studying the course, do not receive permission from the academic advisor to leave the course and do not meet the academic obligations of the course as published by the lecturer at the start of the semester. A student's final grade for such a course shall be zero.

Achievement Evaluation

In order to improve and evaluate students' level of education, students must complete various tasks, including exercises, papers, projects, periodic exams and final exams.

Academic tasks required for each course will be published in the course syllabus handed out by lecturers at the start of courses.

Students will **not be allowed** to take course final exams if they have failed to meet all the compulsory requirements set for exam eligibility. Students who are not eligible to take the exam will be notified of this. Student not eligible to take final exams will be considered to have dropped the course (i.e. their final grade in the course will be zero).

In order to receive a **passing** grade in a course in which a final exam is taken, students must obtain a **passing** grade in the **final exam**. The final grade (which includes the weighing of additional assignments as defined in the course syllabus) will only be calculated for students who received a passing grade in the final exam. The course grade for students who have not passed the final exam will be the exam grade.

Exemption from submitting assignments and projects, due to military reserve duty or illness, will be provided by the course lecturer after receiving written confirmation. Exceptional exemptions will only be granted after a written request from the student to the lecturer. The lecturer must submit the application together with his recommendations for approval to the department head.

Final Exams and Course Passing Grades

Passing grade for final exams

- Passing grade on a final exam in a BSc course is 55.
- Passing grade on a final exam in a MSc course is 60.

Passing grades for courses

- Passing grade for a BSc course is 55.
- Passing grade for a MSc course is 60.

Comment: Students enrolled in an undergraduate degree program and taking a course from a graduate degree program, will be required to receive a passing grade of 60 on the exam and a passing grade of 60 on the course, as is a student enrolled in a graduate program.

Eligibility for a makeup exam

Students are entitled to a makeup exam according to the rules defined in the College's academic regulations.

In addition: Students will be eligible for a makeup (in one course, at most) after failing both exam dates for a course, if the course constitutes the last requirement for completion of their degree (all other courses have been completed on the first or second dates).

Finality of Course Grade and Annual Grade

Course grades will be considered final without appeal to the course lecturer, after two months have passed since the last assignment was submitted.

Grades of all courses taken in a particular academic year shall be considered final with no right to appeal to the Head of Department after December 31 of the calendar year in which course was completed.

Submission Procedures	Submission Dates	Assignment Return and Grading Procedure	Appeal Procedure
Students will keep a copy of submitted papers so they may be graded in the event the original is lost. _____ In the event of non-submission, the grade is weighted as zero. _____ In courses where the teaching language is not Hebrew, assignments shall be submitted in the language in which the course is taught. _____ In courses taught in Hebrew, assignments shall be submitted in Hebrew. Students wishing to submit an assignment in a language other than Hebrew must obtain the approval of the course lecturer and Head of Department _____ Lecturers may invite students to appear before them or before two lecturers to examine students' knowledge and ascertain the paper was written by them. The lecturer will then make a decision accordingly regarding the assignment and its grade. _____ Submissions shall be via the Moodle system only.	**End of course papers:** Until the end of second exam dates. **Seminar papers:** **Final projects / seminars** will be submitted until one month after end of second exam dates. Students not submitting on time will have 0 marked on their chart. Students will be required to re-enroll in the course the following year and meet all course requirements.	Lecturers will assign **grades** for the final project within 45 days of the final date for its submission. Lecturers must **return** checked **thesis papers** within three weeks of their submission. **Seminar papers:** Lecturers will assign **grades** for final seminar papers within 45 days of the final date of submission. Lecturers must **return** checked **papers** within 45 days of submission. Assignments submitted over the summer vacation or up until two weeks before the end of semester 2 – grades shall be delivered within 90 days of submission.	Students may appeal an assignment grade, including seminar papers, if its weight in the final grade is 60% of the final course grade or higher. Appeals shall be submitted in writing within **14 days** of publication of graded papers, detailing the grounds for appeal. Students appealing a grade will be aware the entire paper will be regraded and not only the sections referred to in the appeal. Lecturers may assign a lower grade. The amended grade or lecturers' written decision to dismiss the appeal, will be delivered within **14 working days**. In exceptional cases, the head of school is empowered to decide whether the paper should be passed on to a second examiner.

Progression through the Programs

This section replaces the “Transition from Year to Year” section of the College Regulations.

At the end of each academic year, the academic progress of each student will be reviewed. The evaluation will address the following aspects:

- The weighted annual average of all courses in which the student was enrolled in the academic year in question.
- The weighted annual average of all courses the student passed in the academic year in question.
- The number of credits the student obtained (in passed courses), considering the program in which the student was enrolled in the academic year in question.

The academic status of a BSc student with an annual average, as defined above, below 65 or one who has not obtained a high enough number of credits will be reviewed by the Department’s BSc Pedagogical Monitoring Committee. The academic status of an MSc student with an annual average, as defined above, below 75 or one who has not obtained a high enough number of credits will be reviewed by the Department’s MSc Pedagogical Monitoring Committee.

The Department's Pedagogical Monitoring Committee may make be one of two decisions: cessation of studies or continuation of studies following a curriculum dictated by the committee. A dictated curriculum may include repeating courses the student has passed. A dictated curriculum may also include repeating the entire study program in which the student was enrolled in the year in question (i.e. repeating an academic year).

Prolonged Absence during the Academic Year

Students who require an extended absence during the academic year are required to discuss this with the Department Head in advance or at the earliest opportunity, in order to allow for adjustments to their program and to their obligations in the various courses. Justified reasons for prolonged absence include: particularly prolonged military reserve service, prolonged illness, or other unusual personal reasons at the discretion of the Department Head.

Completion of Studies and Eligibility for a BSc Degree

Students who have completed all required courses listed in the program set by the Department’s Teaching Committee, will be academically eligible for a BSc degree in Computer Science. A degree cannot be obtained after more than eight years from the start date of a student's studies.

Completion of Studies and Eligibility for an MSc Degree

Students who have completed all required courses listed in the program set by the Teaching Committee of the Master's Degree, successfully passed all exams and assignments of the various courses in the program, submitted their final project and received a passing grade, and passed the program's qualifying exam – will be academically eligible for a MSc degree in Computer Science. A degree cannot be obtained after more than four years from the start date of a student's studies.

Using Computer Resources

The use of the College's computer resources, including computers, networks, communications equipment, hardware, software and files (hereinafter "Computer") is subject to the following conditions:

1. Computers are to be used for academic purposes only and not for any other purpose.
2. Credentials for using computers or personal passwords (hereinafter – the “code”) can only be used by students. Codes must be kept confidential and are not to be revealed to any other person. Students are personally responsible for any other use made of their codes.
3. Computers may only be used with the codes given to students and not with any other code and/or account.
4. It is strictly forbidden to use another person's code, eavesdrop on data communication lines or in any way connect with other people's computing resources.
5. Students must use computers in accordance with provisions of the law and College procedures. The instructions of the College's authorized officials must be heeded to avoid any action, act or omission that could cause damage to the computer, data or information stored therein.
6. It is prohibited to use software that has been illegally copied and/or obtained by the user in any way that violates copyright law in College computers.
7. The College does not bear liability towards students in any way for information, software, data and/or anything resulting from use of the Computer and shall not be liable and/or responsible for any loss which may be incurred by the student as a result of using the Computer.
8. College Computers are not to be used for hacking into other systems and/or for gaining illegal access to them.
9. Violation of this obligation constitutes a disciplinary offense and may even constitute a criminal offense.

IV. Course Outlines

BSc Studies

BSc First Year

Introduction to Computer Science	**Course code: 10204011**	**5 credits**
Prerequisites:	**None**	**Year 1 – Semester 1**
Lecture: **Tutorial + Workshop:**	**Dr. Yoram Biberman, Rachel Kahana-Shapira, Tzvi Melamed, Dr. Dvora Ross** **Micha Brenig, Amitai Ben-Nun, Nurit Keratin**	**4 weekly hours** **2+3 weekly hours**

Programming studies at Hadassah College evolve in parallel with the evolution of the programming world: starting with basic programming, moving on to procedural programming, from that to modular, and finally to object-oriented programming.

In this course, we will get to know the basics of programming, and the procedural approach as expressed in C/C++ languages. In addition, as an introductory course to Computer Science as a scientific discipline, we get exposed to issues of it, like: What is an algorithm? How is its effectiveness evaluated? How to write computer programs properly? How does the operating system manage a program's allocated memory? We touch on all of these issues as we walk the programming path: each question will be presented in the context of programs to which it is relevant.

Topics covered: Basic commands (output input [cin] and [cout]). Control statements (conditions and loops). Arrays. Constants and enumerated data types. Functions and their use for writing procedural programs (including value and reference parameters, and value-returning functions). Programming using recursion. Files. Testing the correctness of a program. Familiarity with a number of basic algorithms (bubble sort, insertion sort, quicksort, merge sort, serial search, binary search, Towers of Hanoi, Eight queens puzzle). Intuitive runtime estimation. How the OS allocates memory to a program (on the stack).

Digital Systems	**Course code: 10203011**	**4 credits**
Prerequisites:	**None**	**Year 1 – Semester 1**
Lecture:	**Dr. Simcha Rozen, Dr. Simcha Rozen, David Cohen**	**4 weekly hours**

How is data stored on a computer? How does a computer execute basic operations? In this course we will get to know the fundamental building blocks of computerized systems. We will learn how data is presented in binary systems and will learn about the basic logic gates that allow us to process information of any kind: numbers, letters, music, pictures etc.
The following topics will be covered: Binary numbers and binary systems- number-base conversions, complements. Logic gates. Boolean algebra and basic theorems. Boolean functions: canonical forms, standard forms. Combinatorial logic: SSI, MSI, LSI, arithmetic units. Serial logic: memory and scheduling units.

Discrete Mathematics		**Course code: 10202011**	**4 credits**
Prerequisites:	**None**		**Year 1 – Semester 1**
Lecture: **Tutorial:**	**Dr. Eran London, Dr. Malka Rozenthal, Dr. Idan Telshir** **Elad Chasin, Hadasa Sharvit**		**3 weekly hours** **2 weekly hours**

The course begins with the fundamentals of the language of mathematics. It presents the rules of the game including basic concepts for mathematical studies in general and in particular for theoretical computer science.

The first chapter, **Mathematical Logic**, clarifies what a mathematical proposition is and what a mathematical theorem is. What is a mathematical proof, and what are the possible ways to construct a proof? The chapter includes the following subjects: preface, propositional calculus, logical connectives, logical equivalence, tautology and contradictions, proof by contradiction, complete sets of connectives, DNF, and CNF propositions. Predicate calculus, logical quantifiers, logical equivalence.
The second chapter, **Set Theory**, deals with the central concepts of the course, and various operations on and between sets. At first, we discover that not every expression defines a set (the Barber Paradox). Later we deal with basic concepts and basic operations (set, element, subset, powerset, universal set, intersection, union, complement, symmetric difference, Cartesian product). We represent sets using graphic tools such as Venn diagrams.
The chapter on **Binary relations**, opens a window to several main directions in the studies of mathematics: It allows the definition of an equivalence relation, an ordered set, (and from it the concept of induction) and a function (a graph of a function, range, domain, injective function, surjective function, function composition, inverse function, permutations, characteristic functions.) Those concepts will enable us to define the size of the set.
In the chapter on **Combinatorics**, we will deal with the sizes of sets. We will understand how a child learns to count (and to calculate the size of a set), and following him, we will also do so. We will meet the pigeonhole principle and calculate the number of ways the coat-check attendant in the theater cloakroom can return coats to a group of persons so that not even one member of the group will not return home with his own coat (counting methods, binomial theorem, combinatorial identities, Inclusion–exclusion principle.)

Mathematical Tools for Computer Science (Introduction to Analysis)		**Course code: 10201021**	**4 credits**
Prerequisites:	**None**		**Year 1 – Semester 1**
Lecture: **Tutorial:**	**Dr. Laure Barthel, Dr. Freda Rybnikova, Dr. Nissim Harel** **Dr. Dvora Ross, Leah Butt, David Spindle**		**3 weekly hours** **2 weekly hours**

This is the first course in a series of math courses. It focuses on the mathematical tools required to describe geometric objects in the plane and in the space and the basic concepts of functions.
Analytical geometry in the plane and in the space: vectors, dot product, cross product, lines and planes. Basic equations and properties of conic sections. Complex numbers: basic definitions and properties, the Gaussian plane, polar representation, roots of unity. Functions: basic properties, graph, translation of a graph, composition of functions. Limits of functions: intuitive definition using numerical and graphic examples, computation of limits of rational functions using polynomial division. Derivative: definition, meaning and differentiation rules.

Linear Algebra A		Course code: 10201071	4 credits
Prerequisites:	**None**		**Year 1 – Semester 1**
Lecture: **Tutorial:**	**Dr. Arie Yakir, Dr. Dvora Cohen, Dr. Arie Yakir** **Dr. Dvora Cohen, Dr. Dvora Cohen**		**3 weekly hours** **2 weekly hours**

This is a basic math course that deals first with systems of linear equations with any number of variables. The material taught in this course is essential for further courses in mathematics and computer science. We shall learn how to use basic mathematical tools such as matrices, coordinates, transformations (reflection, rotations, projections and more).

Topics covered: Fields. Systems of linear equations over a field. Matrices. Elementary operations, elementary matrices, and matrix multiplication over a field and over a ring. The determinant of a matrix. Vector spaces: subspaces, bases, dimension. Row space, column space and null space of a matrix. Coordinate matrix. Linear maps: kernel and image, arithmetic of linear transformations. Representations of linear transformations by matrices, properties of the representation. Lagrange interpolation. The determinant as a volume function.

Mathematics Workshop A		Course code: 10208021	No credits
Prerequisites:	**None**		**Year 1 – Semester 1**
Lecture:	**Oded Gutman, Leah Butt, Tuval Coleman**		**2 weekly hours**

The workshop helps students cope with the nature of post high school math. Highlights in the workshop are the concept of mathematical proof, the correct use of mathematical language, and various problem-solving techniques.

Modular Programming		Course code: 10208021	5 credits
Prerequisites:	**Introduction to Computer Science**		**Year 1 – Semester 2**
Lecture: **Tutorial + Workshop:**	**Dr. Yoram Biberman, Dr. Dvora Ross, Tzvi Melamed** **Micha Brenig, Amitai Ben-Nun, Nurit Keratin, Micha Brenig**		**4 weekly hours** **2+3 weekly hours**

This course directly follows 'Introduction to Computer Science'. As such, it follows the programmatic evolutionary journey on to the modular paradigm, and to the very edge of object-oriented programming. The course touches on computer science as a form of science by discussing various data structures: especially lists and trees (from a programming perspective) and their effectiveness; and classical algorithms. The course elaborates on the role of pointers in the C programming language (for defining dynamic arrays, lists, trees, and generic, polymorphic programming). The course elaborates on “proper programming”, including on program testing, and writing libraries.

Topics covered: Handling external files (opening/closing for reading/writing, get/put pointers, reading and writing on the same file simultaneously). Dynamic array allocation (including: definition, dynamic memory allocation, memory release, pointer arithmetic, differences between static and dynamic memory allocation, transferring/returning pointers from functions, pointer to pointer) structures. Linked lists (including merge sort of lists, handling lists through pointer to pointer). Binary trees (mainly binary search trees, including insertion, search, deletion, and various tree algorithms). Functions pointers (and writing callback functions). Generic (void *) pointers and their use for writing polymorph programs in C language (also with function

pointers). Program division into files, statement versus definition, preprocessor guidelines, and use of makefile. The compilation and linking process in C. Namespaces. Handling exceptions. Various topics: C++ strings, inline functions, function overloading, assert, sizeof, typedef Libraries: standard C libraries, build and add, static and dynamic library. Library use. Software testing methods (such as valgrind). scanf/printf, argc/argv. Template functions. Linux programming tools, including debuggers.

Hardware Systems, Software & Programming in Assembly Language	**Course code: 10203021**	**4 credits**
Prerequisites:	**Digital Systems**	**Year 1 – Semester 2**
Lecture:	**David Cohen, Dr. Ayelet Goldstein, Alba Slavin, David Cohen**	**4 weekly hours**

When speed is important to the success of the system – in games, film processing, medical equipment, robots – matching software to computer hardware is a critical component of design. In this course we will learn how to relate to the PC using its own language and know how to intervene between C language programming and running the software on the hardware.

Topics studies: Introduction to computer organization and microprocessors, Von Neumann structure, software hierarchy, command structure and machine language, accumulators, methods of addressing, stacking, interruptions, memory management. 8086 processor review: assembly language programming, machine language translation, running program structure. Operating system services: keyboard reading, on-screen viewing, file access, loading and running programs. Analysis of C language programs after compilation: data frame, dynamic variables, functions, parameter transfer, recursion. Review of IA-32 (Intel 32/64-bit Processor Structure), 32-bit assembly programming in Linux environment, connecting assembly language functions to C language programs.

Introduction to Theoretical Computer Science (Discrete Mathematics B)	**Course code: 10202021**	**4 credits**
Prerequisites:	**Discrete Mathematics**	**Year 1 – Semester 2**
Lecture: **Tutorial:**	**Dr. Eran London, Malka Rozenthal, Dr. Idan Telshir** **Elad Chasin, Ayelet Amsalem**	**3 weekly hours** **2 weekly hours**

The course includes several different chapters dealing with basic aspects of modern computer science. The course offers students a glimpse of the various fields and builds a solid foundation for their studies.

1. **The concept of infinity**. What does it mean? Is there more than one infinity? Is there a “smallest” infinity? Does every infinity have a “larger” infinity? During discussions we will introduce the following concepts: infinite sets, countable sets, Cantor's diagonal argument, the cardinality of a power set (Cantor's theorem), Bernstein-Schroder's theorem.
2. **Graph theory**. We will present various problems using graphs and try and solve them in a generalized way. Some of the questions we will cover: How can the world map be clearly displayed using only a few colors? What bothered Euler when he went out for a walk in his city? How can you help the younger generation in the matchmaking world? Can the Internet ensure that messages are delivered quickly, and how? How many roads does Jerusalem need to maintain so it is possible to drive from the city center to every intersection in town? The technical concepts we will encounter include the following: definitions (vertex, edge, simple graph, directed graph, path, circle, simple circle, polygon), connected component, connectivity, tree, forest, number of edges and vertices in a tree, planar graph, Euler formula for planar graphs, graph coloring, coloring number, matching, Hall's marriage theorem.
3. **Introduction to Discrete Probability Theory**. We will understand how probability theory connects to day-to-day operations: how to design a poll and what is its credibility? How can one play poker over the Internet? Can a coin be tossed fairly when the partner cheats? Is a student with a higher grade point

average than another student also a better student? How can we ensure that the Internet does not crash when several lines of communication are torn? We will meet the following technical concepts: discrete probability spaces, independent events and conditional probability, random variables and expected value, distribution and variance, basic inequalities.

4. **Growth rate of functions**. How do you determine that one algorithm is more effective than another? What can you do with your computer? Is it enough to be a good programmer to solve any problem? We define and deal with the following concepts: order of magnitude of functions and sequences, asymptotic behavior, approximate solution of recurrence formulas.

Calculus: Single Variable Calculus	**Course code: 10201031**	**4 credits**
Prerequisites:	**Mathematical Tools for Computer Science**	**Year 1 – Semester 2**
Lecture: **Tutorial:**	**Ronit Katz, Dr. Freda Rybnikova, Dr. Nissim Harel** **Dr. Dvora Ross, Shira Shvets**	**3 weekly hours** **2 weekly hours**

The course introduces students to the core ideas of analysis and shows how they can explore the properties of functions.

Real numbers: Properties of real numbers, intervals, absolute value, upper bound, lower bound. Comparison of the number concept in math and programming.

Limits of a function: How to ensure that the output of a function is within the desired range while controlling input? We will see how the formal definition of the bound is the mathematical answer to this computational problem.

Continuous functions and derivative of a function: We will see what can be learned about a function from its limits and derivatives. Intermediate Value Theorem and Weierstrass theorem. Rolle's Theorem, Lagrange's mean value theorem, study of functions (intervals of increase and decrease, extrema, convexity, asymptotes), L'Hôpital's rule. Monotonic functions. Theorems about inverse functions, inverse trigonometric functions, the logarithm function and the exponential function).

Integrals: Definite integrals, indefinite integrals and the relationships between them.

Linear Algebra B	**Course code: 10201081**	**4 credits**
Prerequisites:	**Linear Algebra A**	**Year 1 – Semester 2**
Lecture: **Tutorial:**	**Dr. Arie Yakir, Dr. Dvora Cohen, Dr. Arie Yakir** **Dr. Dvora Cohen, Dr. Dvora Cohen**	**3 weekly hours** **2 weekly hours**

This course is a continuation of Linear Algebra A. Topics covered in this course: Eigenvalues and eigenvectors. Diagonalizable matrices and operators. Inner product spaces over the Real Field and over the Complex Field. Orthonormal bases and the Gram-Schmidt orthogonalization process. Geometry of inner product spaces. Fourier coefficients. Best approximation in finitely generated subspace. Orthogonal matrices and orthogonal transformations. A short introduction to affine geometry.

Mathematics Workshop B		**Course code: 10208031**	**No credits**
Prerequisites:	**Mathematics Workshop A**		**Year 1 – Semester 2**
Lecture:	**Oded Gutman, Hadasa Sharvit**		**2 weekly hours**

This workshop helps students cope with the nature of post high school math. The workshop covers the concept of mathematical proof, correct use of mathematical language, and various problem-solving techniques.

BSc Second Year

Introduction to Object Oriented Programming & Software Engineering	**Course code: 10204032**	**5 credits**
Prerequisites:	**Introduction to Computer Science, Modular Programming, Data Structures (participation)**	**Year 2 – Semester 1**
Lecture: **Tutorial + Workshop:**	**Michal Alhanaty, Rachel Shapira, Zvi Melamed** **Yehezkel Bernat, Amitai Ben-Noon, Matan Perel, Ruti Bornstein**	**4 weekly hours** **2+2 weekly hours**

How are large-scale and complex applications developed? How do you make the applications complete and general? What is the key to clear, convenient and easy to maintain programming? Answers to these questions are the basis for Object Oriented Programming & Software Engineering. The course will study the principles while applying them in C++. The course will be accompanied by examples and exercises of complete applications with their variety of components: data structures, algorithms, artificial intelligence, interface, graphics, information security, performance and more. The acquisition of principles, this semester, will be mainly around the tools of classes, inheritance, polymorphism, UML diagrams, and introduction to design patterns.

System Programming & Introduction to Parallel Programming	**Course code: 10203052**	**5 credits**
Prerequisites:	**Introduction to Computer Science, Modular Programming, Hardware Systems, Software & Programming in Assembly Language**	**Year 2 – Semester 1**
Lecture: **Tutorial + Workshop:**	**Dr. Yoram Biberman, Michal Goldstein, Dr. Udi Conly** **Tamar Bash, Tamar Bash, David Spindle**	**4 weekly hours** **2+2 weekly hours**

How is it possible to write a program that reads data from two sources at the same time? In general: performs several tasks in parallel and in a coordinated manner? How can we make good use of multiple processors?

The course includes two parts: a programming component that presents the Unix's system calls, and inter-process communication (IPC) tools, and a theoretical component that discusses the basics of operating systems, especially processes and threads.

Topics covered: Introduction to operating systems. Topics in architecture relevant to operating systems (in particular an interrupt). Processes (including process creation in Unix: fork(), exec()). Communication between processes in Unix (signal, pipes, named pipes, message queues, sockets, shared memory). Threads (including the pthreads library). Processor scheduling (especially on Linux). Process synchronization (especially semaphore in theory, and in Unix and pthread). Mutual exclusion. Parallel programming paradigm: options and challenges.

Data Structures	Course code: 10202032	4 credits
Prerequisites:	**Linear Algebra A + B, Discrete Mathematics, Introduction to Theoretical Computer Science**	**Year 2 – Semester 1**
Lecture: **Tutorial:**	**Dr. Eran London, Dr. Gili Scholl, Efi Naftali** **Chaya Zilberman, Esther Meisel, Moshe Munk**	**3 weekly hours** **2 weekly hours**

We will work on ways to effectively represent information on the computer so that solving problems is fast and cost-effective. We will encounter stacks, queues, linked lists, rooted trees, heaps and priority queues. The course will cover the following topics:

- Sorting issue: How to sort a data set conveniently and quickly? We will encounter various (and often strange) solutions that address different problems: insertion sort, selection sort, heapsort, mergesort and quicksort. We will encounter lower bounds for sorting. We will deal with the expected running time of quicksort. We will conclude with linear sorts: counting sort, radix sort and bucket sort.
- Dynamic data structure search problem: How can a ("huge") phonebook be presented on a computer, be continuously updated and searched quickly? We will meet binary search trees, especially balanced trees (such as a red-black tree).
- What is the appropriate data structure for managing a "small" pool of unexpected items (such as rooms in a hotel chain that should be able to accommodate every possible visitor)? We will study hash tables and hash functions. We will see how a whole encyclopedia can be stored on DVD so that the search time for each entry is very short.
- What is the most effective way to compress data so that it does not "take up" a lot of memory space? This was answered by Huffman codes.
- What is the most convenient way to travel in a foreign city so that we learn it perfectly, and how does it relate to the wave movement in a lake and solving mazes? We will answer this through breadth first and depth first searches in graphs.
- How can a minimal investment in road construction be ensured without interfering with the flow of traffic between any two points in the city?
- How can the distance between any two cities in the country be calculated, given that we are only given the lengths of roads between adjacent intersections? How does this relate to the international currency market?

Infinitesimal Calculus: Integral Uses & Approximate Calculations	Course code: 10201042	4 credits
Prerequisites:	**Infinitesimal Calculus: Single Variable Calculus**	**Year 2 – Semester 1**
Lecture: **Tutorial:**	**Dr. Laure Barthel, Dr. Malka Rozenthal, Dmitry Goldstein** **Hadasa Sharvit, Boris Kanevsky**	**3 weekly hours** **2 weekly hours**

Integral Uses: With integrals you can calculate many geometric data, such as the area between curves or the volume of a solid. We will also see when the integral can be generalized to infinite cases.
Area and volume calculations, volume of solids of revolution. Improper integrals.
Approximate Calculations: How can a computer calculate *sin x* if it only knows how to add and multiply? How do we calculate the value at which a function vanishes if we do not have a formula? The second part of the course addresses such questions. We will use graphic and numerical software to illustrate the concepts.
Sequences: definition, limits , Cauchy sequence, sequences defined by a recurrence relation. Series: definition, convergence (absolute and conditional), convergence tests, Leibniz series. Taylor polynomial. Taylor series of function and convergence to the function. Power series: definition, radius of convergence,

interval of convergence, differentiation and integration of power series, representation of elementary functions as power series, application of power series to the computation of approximations, and the solving of differential equations. Numerical analysis: bisection method, the Newton–Raphson method, iterative method, Newton–Cotes numerical integration (the trapezoid rule, the Simpson's rule).

Algebraic Structures	**Course code: 10201092**	**4 credits**
Prerequisites:	**Mathematical Tools for Computer Science, Linear Algebra A + B**	**Year 2 – Semester 1**
Lecture:	**Ronit Katz, Ronit Katz, Dr. Arie Yakir**	**4 weekly hours**

In this course we will study mathematical topics that are needed in computer science, in particular for the study of algorithms, cryptology, and error correcting codes. Topics covered: Groups: definition of a group and a subgroup, the symmetric group, homomorphism and isomorphism, permutation representation of a group, orbits, right and left cosets, quotient group, the fundamental theorem of homomorphism. Rings: definition of a ring and a subring, ideal, integral domain, field, rings of polynomials, Euclidean rings, principal ideal domain, quotient ring, construction of finite fields. Number theory: the extended Euclidean algorithm, congruence. Fermat theorem, Euler theorem, Wilson theorem, Chinese remainder theorem, quadratic residues, discrete Fourier transform.

Scientific Writing & Presentation – Group 1	**Course code: 10206252**	**2 credits**
Prerequisites:	**None**	**Year 2 – Semester 1**
Lecture:	**Sofie Chazanov, Dr. Simcha Rozen, Dr. Ariel Furstenberg**	**2 weekly hours**

Technical and scientific writing has become an important skill for computer science students and graduates. The aim of the course is to develop scientific writing and scientific presentation skills. The following topics will be covered: reading academic papers in computer science, characterizing of scientific writing, active reading. The structure of the scientific paper and the principles of writing a summary of a paper. Techniques of effective search in databases and search engines. Preparing a written academic presentation. During the course we will practice these various skills by writing reports, selecting a paper and presenting it to the class.

Object Oriented Programming and Game Development	**Course code: 10204042**	**5 credits**
Prerequisites:	**Introduction to Object Oriented Programming & Software Engineering**	**Year 2 – Semester 2**
Lecture: **Tutorial + Workshop:**	**Michal Alhanaty, Rachel Shapira, Zvi Melamed** **Yehezkel Bernat, Amitai Ben-Noon, Matan Perel, Ruti Kleinman**	**4 weekly hours** **2+2 weekly hours**

Pac-Man, Digger, GTA (Grand Theft Auto), Lemmings and many other computer games are examples of worlds with objects, actions, and interactions. Applications of this type clearly demonstrate the ideas, principles and challenges that exist in object-oriented programming. The course will cover advanced object-oriented programming topics such as: templates, iterators, use of existing libraries and design patterns. The programming experience will include applications from the world of computer games based on data structures and algorithms taught in theoretical courses.

Scripting Operating Systems & Programming	Course code: 10203062	5 credits
Prerequisites:	**Introduction to Computer Science, Modular Programming, Hardware Systems, Software & Programming in Assembly Language, System Programming & Introduction to Parallel Programming**	**Year 2 – Semester 2**
Lecture: **Tutorial + Workshop:**	**Tamar Bash, Michal Goldstein, Dr. Ehud Conley** **Ruti Bornstein, David Spindle**	**4 weekly hours** **2+2 weekly hours**

How does the operating system manage main memory? How is disk organized? What is a directory? How can you make sure that even if the disk crashes the contents of the files will not be lost? What happens when mapping a file to memory? What happens when we turn on the computer? The course will answer these and other questions.

Studies topics: Main memory organization (including: swapping, paging, segmentation, demand-paging). File system (including NFS, log-based file system). Disk management and scheduling. Management of swap area. Caching and buffering. RAID organization. Input/output systems (theoretical and Unix). Unix file system management (including: device nodes, soft\hard link) Unix file and directory handling: open(), creat(), read(), write(), lseek(), fcntl() in particular to lock files and handle status flags, umask(), access(), chmod(), stat(), rename(), unlink(), opendir(), readdir(), mkdir(). Memory mapped files: mmap(). Familiarity with shell programming using Python language. Methods for dealing with mutual exclusions (Banker's algorithm, detection recovery from mutual exclusion) processor scheduling in theory and Linux. Introducing cygwin software.

Algorithms	Course code: 10202042	4 credits
Prerequisites:	**Linear Algebra A + B, Discrete Mathematics, Introduction to Theoretical Computer Science, Algebraic Structures, Data Structures**	**Year 2 – Semester 2**
Lecture: **Tutorial:**	**Dr. Eran London, Dr. Hadasa Yakobovich, Efi Naftali** **Haya Zilberman, Esther Meisel, Moshe Munk**	**3 weekly hours** **2 weekly hours**

We will deal with many practical problems from all areas of computer science and find out how to approach them and to solve them. We encounter broad fields and topics. The solutions are based on all the mathematics and theory taught in previous courses.
Among the topics we will address are the following:

- What is an algorithm? What are the resources and prices (time, place)?
- What do we mean by "divide and conquer"? How does one solve a problem by dividing it into "smaller" problems?
- How to quickly identify a short word in a huge file? (Fast Fourier Transform and its applications).
- Which sculptures will Ali Baba choose when he breaks into the museum? (dynamic programming, knapsack problem).
- Should we be greedy and when?
- How do you route trucks on a road network and how does this relate to Hall's marriage theorem and solving Sudoku? (flows in graphs).
- What is modern cryptography? Why do we agree to provide our credit card information over the internet where it is visible to all and why did our ancestors not do so? (RSA cryptosystem, Rabin cryptosystem).
- In what way is a gambler preferable on a standard "solid person"? How does randomization help solve difficult problems?

Infinitesimal Calculus: Curves & Surfaces	**Course code: 10201052**	**4 credits**
Prerequisites:	**Linear Algebra A + B, Infinitesimal Calculus: Integral Uses & Approximate Calculations**	**Year 2 – Semester 2**
Lecture: **Practice:**	**Dr. Laure Barthel, Dr. Malka Rozenthal, Dmitry Goldstein** **Ronit Katz, Ayelet Amsalem, Boris Kanevsky**	**3 weekly hours** **2 weekly hours**

In this course we will learn how the ideas of analysis can be used to study curves and surfaces defined by parametric equations or by equations. Graphic software is widely used to illustrate the concepts.
Topics covered: Vector functions: smooth parametrization, tangent vector, normal vector, binormal vector, re-parametrization, arc length, curvature. Functions of several variables: limits and continuity, partial derivatives, directional derivatives, differentiability, chain rule, gradient, maxima and minima, Lagrange multipliers. Curves and Surfaces: definition by parametrization or equation, tangent line or plane. Multiple integrals: computation, Fubini's theorem, change of variables in double integrals (mainly to polar coordinates).

Probability Theory	**Course code: 10201102**	**4 credits**
Prerequisites:	**Mathematical Tools for Computer Science, Calculus: Integral Uses & Approximate Calculations, Calculus: Curves & Surfaces (can be studied simultaneously), Discrete Mathematics**	**Year 2 – Semester 2**
Lecture:	**Dr. Ayelet Goldstein, Dr. Ayelet Goldstein, Shmuel Dahan**	**4 weekly hours**

In recent years, understanding the laws of probability theory has become essential for the development of fast and efficient algorithms. Probability algorithms, that is, algorithms that make random choices during their run, prove to be a cost-effective way to solve problems that would otherwise not be solvable in a reasonable time. In this course we will discuss the basics of probability theory and learn how to make calculations and assessments under conditions of uncertainty.
Topics covered: repeat of discrete uniform distribution: probability space. Conditional probability. Bayes formula. Independence. A family of independent events. Probability mass function of random variable, cumulative distribution function. Expected value and variance of a random variable. Mathematical properties. Expected value of a random variable function. Special discrete distributions: Bernoulli, Binomial, Geometric, Discrete, Hypergeometric, and Poisson random variables. Continuous random variables: probability density function of a continuous random variable. Cumulative distribution function. Variance and expected value. Special distributions: continuous uniform, exponential and normal. Discrete and uniform two-dimensional random variable: joint probability function of a pair of random variables, marginal distribution, conditional distribution. Conditional expected value as a random variable. Law of total expectation and its applications. Distribution of functions of random variables: sum distribution. Probability inequalities. Expected value and variance of sum. Independent and uncorrelated random variables. Central bound theorem. Advanced topics: entropy of a random variable. Stochastic processes. Markov chains and their applications in computer science.

Scientific Writing & Presentation – Group 2		Course code: 10206252	2 credits
Prerequisites:	**None**		**Year 2 – Semester 2**
Lecture:	**Sofie Chazanov**		**2 weekly hours**

Technical and scientific writing has become an integral part of the professional world of computer science. The aim of the course is to develop writing and presentation skills required during both the degree and professional life after graduation. The course will focus on the following topics: reading articles in the field of computer science, characterizing scientific writing, dealing with article reading and turning reading into active learning. The structure of the scientific paper, the principles of summary and academic experience. Informed searches in databases and search engines. Presentation construction. During the course we will practice these various skills by writing reports, selecting an article and presenting it to the class.

BSc - Third Year

BSc - Third Year – Mandatory Studies

Internet Programming A	**Course code: 10204053**	**4 credits**
Prerequisites:	**Introduction to Object Oriented Programming & Software Engineering, Object Oriented Programming & Game Development**	**Year 3 – Semester 1**
Lecture:	**Dr. Solange Karsenty, Lior Grosman**	**4 weekly hours**

The course provides knowledge for the development of websites, with an emphasis on user friendly, interactive and responsive websites. The course covers mainly client-side technology, but also server side-technology: markup languages, scripting languages, network protocols, interactive graphics, event-driven programming, and databases.
On the client-side this includes: HTML5, CSS3, JavaScript and libraries such as Twitter Bootstrap or React. On the server-side this includes HTTP and networking, database management, building components such as user registration and authentication, validation, dynamic content management, database search, shopping carts. We will learn to develop systems based on the Model-View-Controller architecture, using frameworks and libraries such as NodeJS.

Computer Networking	**Course code: 10203083**	**4 credits**
Prerequisites:	**System Programming & Introduction to Parallel Programming, Scripting Operating Systems & Programming**	**Year 3 – Semester 1**
Lecture:	**Amiel Lieber, Avigail Gertie, Amiel Lieber**	**4 weekly hours**

Using the Internet is a daily part of our lives, and programming in a networked environment is a basic skill in software system development. Proper use of network infrastructure is an important factor in creating a successful experience. In this course, we will explore the essentials of computer networking and the integration of data transfer systems, voice calls, and multimedia.
Topics covered: Basic concepts in communication and open network models, network applications, management of information transfer, communication between different programs, reliability and end-to-end quality of service, routing of messages in heterogeneous networks, local networks, wired and wireless physical communication. In each chapter, we will highlight the practical methods and protocols of Internet communication and local networks.

Computer Architecture	**Course code: 10203072**	**3 credits**
Prerequisites:	**Hardware Systems, Software & Programming in Assembly Language**	**Year 3 – Semester 1**
Lecture:	**Dr. Martin Land, Dr. Ayelet Goldstein**	**3 weekly hours**

Architecture is the field of computer science that provides a platform for practical realization of innovations in programming and algorithms. The huge increase in the number of lines of code in today's complex software systems is only possible thanks to a parallel increase in hardware speed.
In this course you will become familiar with the professional techniques in computer architecture and the essentials of existing technologies in modern computers.

Topics covered: Review of computer organization, instruction set architecture, measurements and benchmarks, performance theory and quantitative analysis of methods for executing microprocessor commands, the transition from CISC to RISC methods, pipelining and bound analysis, computer account and ALU planning, methods for managing memory and cache methods, internal communication on the motherboard, superscalar systems, multicore processors, the PC today as realization of architecture achievements.

Automata and Formal Languages	**Course code: 10202053**	**3 credits**
Prerequisites:	**Discrete Mathematics, Introduction to Theoretical Computer Science, Data Structures**	**Year 3 – Semester 1**
Lecture:	**Dr. Faraj Shiban, Dr. Esther David, Dr. Faraj Shiban**	**3 weekly hours**

A formal language is a collection of strings that have a particular trait, or a certain structure, for example, prime numbers, or words of equal length with the same letter at the beginning and end, and the like. Almost every mathematical problem can be presented as a language in a way that solving the problem is reduced to deciding whether a given word is or is not in a given language. In this course we will introduce and study mathematical models of computing machines that can solve such decision problems. We will discuss the computational power of each of these models and characterize the types of languages it can recognize.
We'll start with the model of finite state automaton. This is a simple computer, with finite and restricted memory. Finite state automata capture only a small set of algorithms, but they are useful in simple procedures in string matching algorithms. Languages that can be recognized by such automata are called "regular languages". Later we'll introduce a stronger model, with infinite memory but with limited and conditioned approach to memory, called "pushdown automaton". This model can solve decision problems that cannot be solved by finite state automata, and they are very useful in parsing algorithms. A push down automata has an equivalent counterpart called "context free grammar". This is yet another model of computation, it consists of a finite set of "grammatical rules" of special type, and it represents the language of all strings that can be derived from a "start symbol", using the given rules. The languages that can be "generated" by context free grammars are exactly the languages that can be recognized by pushdown automata, and they are called "context free languages".
In this course we concentrate on two families of languages: regular languages and context free languages. We'll study and become familiar with many structure features of languages in each of these two families, and discuss various decision problems regarding their structure.

Logic for Computer Science	**Course code: 10202073**	**3 credits**
Prerequisites:	**Discrete Mathematics, Introduction to Theoretical Computer Science**	**Year 3 – Semester 1**
Lecture:	**Efi Naftali, Alba Slavin, Dr. Yermiyahu Kaminski**	**3 weekly hours**

What is "proof" in math? When is a mathematical claim said to be "correct"? Can any correct claim be proved? And is what can be proved necessarily true? In this course we will get down to the basics of mathematics, define the most basic concepts like "proof" and "correctness" and understand the relationship between the two.
We'll start with very simple mathematical language (propositional calculus), we will learn to formulate mathematical arguments using "connections" that connect basic claims to more complex ones. We'll see a way to write uniform formulas (normal form), which makes it easier to handle. We will discuss different groups of connections, and when such a group makes it possible to create all possible arguments ("whole group" and "limited group" of connections). We will define what "correctness" is (names and true values) and what "proof" is; we will get to know tools for creating proof (rule of inference and proof from a premise). We will prove that everything that is proven is "correct" (soundness theorem) and that everything that is

correct can be proven (completeness theorem). We will prove the compactness theorem which says that if a claim comes from an infinite set of assumptions, it already results from a finite subset of assumptions. We will understand that there is currently no practical way to decide for each formula if it is correct and we will recognize a sub-family of formulas (Horn) for which this is possible.
The "composite relations" language allows for much more composite mathematical arguments, we will go through the whole process with this language too, talk about correctness versus proof, soundness theorem and completeness theorem (without proof), and compactness theorem. We will learn about Peano axioms and the standard model of natural numbers, and Gödel's incompleteness theorems (unproven) that say that in the above system of natural numbers there will always be something that neither it or its negation can be proven, and that the system cannot be expected to prove what we have proven to be true!
Later we will discuss even more complex languages that allow for relative correctness of time (temporal logic), and languages that extend the concept of right / wrong, and also allow something in the middle.

Internet Programming B	**Course code: 10204153**	**4 credits**
Prerequisites:	**Introduction to Object Oriented Programming & Software Engineering, Object Oriented Programming & Game Development, Internet Programming A**	**Year 3 – Semester 1**
Lecture:	**Dr. Solange Karsenty, Lior Grosman**	**4 weekly hours**

This course focuses on advanced Java technologies for the development of websites, with an emphasis on server-side components such as database management, communication, security, and REST web services.
The course details Java programming principles and technologies for building dynamic websites, network programming, concurrent programming, Servlets and the Spring framework and template engines. Combining technologies with the material taught in the first course (Internet Programming A) results in designing and building complete websites, providing highly interactive pages based on advanced server side components. The course ends with the construction of an mini website that incorporates all the technologies learned.

Databases	**Course code: 10204073**	**4 credits**
Prerequisites:	**Discrete Mathematics, Introduction to Theoretical Computer Science**	**Year 3 – Semester 1**
Lecture:	**Dr. Simcha Rozen, Dr. Simcha Rozen, Dr. Yoram Biberman**	**4 weekly hours**

Large software systems handle a huge amount of data. The systems must be able to retrieve the data efficiently and quickly, as otherwise their performance will be unreasonable. The course presents theoretical models for handling a large amount of data, and their practical applications. For example: How can we retrieve data using different characterizations? How should the system organize the data so it can be retrieved efficiently? How do we make sure that if only one ticket is left on a flight, it will not be sold, at the same time, to two different customers by two travel agents? How do we make sure that even if the computer crushes after a customer withdraws money from an ATM his account will be charged the withdrawal?
Topics covered: Introduction to databases. The Relational model. Relational algebra. The Tuple Relational Calculus. SQL. Integrity constraints and database design (keys, functional dependencies, normal forms: BCNF, 3NF). Entity-Relationship Model. Transaction processing, concurrency control (serializability, conflict serializable schedules and view serializable schedules, protocols ensuring serializability: lock bases protocols (e.g. 2PL) , and timestamps base protocols).

The course makes use of PostgreSQL database management system.

Computability & Computational Complexity	Course code: 10202063	3 credits
Prerequisites:	**Algorithms, Logic for Computer Science, Automata and Formal Languages**	**Year 3 – Semester 1**
Lecture:	**Dr. Faraj Shiban, Dr. Esther David, Dr. Faraj Shiban**	**3 weekly hours**

This course us about studying the limits of what can be done by computers. Even if we have all the time in the world, not everything is doable! In this course we will demonstrate, for example, that no computer program can be written that does the following: give it a computer program A and input X for A, and our program must say if the input X enabled on A has a "bug". In order to be able to prove this we need an exact workable definition of "what is computer" or "what is computation".

"Turing Machine" is the theoretical computer invented by Alan Turing (German Enigma Cipher Breaker) in 1936. It is regarded (by Church Turing Thesis) as a formal definition of "computer" and/or of "computation". A problem is considered "solvable" if there is a TM that solves it, and a function is regarded "computable" if there is a TM that computes it.

The material in this course is divided into three chapters:

Chapter 1. The Church Turing Thesis: this is an introductory chapter, in which we introduce several versions of TM's and prove that the various versions are equivalent in their computational power.

Chapter 2. Decidability: In this chapter we discuss solvability of decision problem by computer, without any consideration of the computational resources needed for the solution (such as running time or size of memory). We discuss three classes of problems: R, RE, and co-RE, and we learn to distinguish between solvable, partially solvable and unsolvable problems. In this chapter we talk about theoretical solvability, without considering the practicality of the solution.

Chapter 3. Tractability: In this chapter we will learn to distinguish between problems that can be solved efficiently (polynomial time complexity, logarithmic space complexity), and problems whose solution requires exponential running time, or exceptionally large memory. We discuss various complexity classes such as P, NP, co-NP, PSPACE, L, NL, and study the interesting phenomenon of NP-completeness, and completeness in other complexity classes. We will see that many of the questions we face here do not currently have an answer, and probably never will.

Final Project	Course code: 10204143	8 credits
Prerequisites:	**Completion of second year mandatory courses**	**Year 3 – Annual**
Lecture:	**Dr. Yoram Yekutieli, Dr. Simcha Rozen, Dr. Yoram Yekutieli** **Prof. Michael Berman** **Prof. Michel Bercovier** **Dr. Michal Alhanaty** **Dr. Ayelet Goldstein** **Dr. Yoram Yekutieli** **Dr. Martin Land** **Dr. Solange Karsenty** **Or Kadrawi** **Avigail Rabin Hefner**	**12 weekly hours**

Have you always dreamed of developing an original app for iPhone or playing with camera-guided robots? The final project is your opportunity to develop advanced applications from concept to implementation. The project is, on the one hand, an opportunity for independent work and, on the other hand, is supervised and accompanied at each stage of planning and execution. As part of the project, you will be able to bring your

capabilities to fruition and design a respectable “entry ticket” into the industry. Project topics will be selected by students in coordination with the project supervisors from a variety of fields such as robotics, advanced user interfaces, internet, graphics, image processing, computer vision, microprocessors and advanced programming. At the planning stage, students will be required to formulate a project proposal and, after approval, will proceed to the project execution phase. The implementation phase will focus on project construction, project review, project documentation, project packaging and project presentation.

Examples of thesis projects from previous years include a cyclist system for Android environment, a camera-guided robot system for security, an application for correcting spelling and grammar errors, Hebrew writing and real-time gesture recognition software, training software to help deaf children practice speaking and more. Some projects are carried out in collaboration with departments of the School of Communication and Design and the Department of Communication Disorders as well as with external parties such as the Department of Forensics, Variety Center for Special Needs Children, The Milbat Association and leading companies in their fields such as Intel, Exodius, Mobileye, and Malam.

Courses marked with * will not be offered in 5780 (2020)

Machine Learning		**3 credits**
Prerequisites:	**Linear Algebra A + B** **Analysis A + B**	**Year 3 BSc, MSc Semester 1**
Lecture:	**Dr. Yoram Yekutieli**	**3 weekly hours**

What is Learning? What are learning systems? Can a machine learn and which concepts are learnable? How can I teach a machine? With what methods and for what problems?

In recent years, it has been demonstrated that not only can machines learn, but that it is an incredibly essential tool for dealing with complicated problems and the congestion of information that is overwhelming us. Automated systems that learn to recognize patterns are now an important component of a huge variety of fields and applications.

In the course we will introduce the field of machine learning and its relation to other areas. We will teach the following topics through practical examples:

Regression: matching model to data. Mean square error methods. Support Vector machines. Detection (Is there a face in the picture? Where?), recognition (What is the object in the input, is this a person?) Identification, individualization (Who is this cat? Is it Mitzi?), supervised learning, classification (Does the input presented belong to one group or another, is it an orange or an apple?). Nearest neighbors algorithm, Naive Bayes classifier, decision trees. Artificial neural networks. Unsupervised learning, clustering, dimensionality reduction, reinforcement learning, probabilistic graphic models, genetic algorithms. Software libraries including Matlab and Python (scikit-learn) tools that implement learning systems.

Artificial Intelligence in Medicine		**3 credits**
Prerequisites:	**Object Oriented Programming & Game Development** **Algebra A + B**	**Year 3 BSc, MSc Semester 1**
Lecture:	**Dr. Ayelet Goldstein**	**3 weekly hours**

The tasks that physicians are required to perform are many and varied, such as patient screening, diagnosis, proper and optimal care selection, data analysis, summary information. These tasks require methodologies from various fields such as computer science, artificial intelligence, statistics and decision analysis.

In this course we will explore the world of medical decision support systems and the different methodologies used in this field.

We will study the following topics: Introduction to medical informatics, medical records and standards, temporal databases, knowledge representation, different types and characteristics of decision support systems: rule-based systems, knowledge-based systems, medical diagnostic systems. Time-dependent inference, human judgment biases, decision trees, Bayesian belief networks, abstraction of time-oriented data, visualization and analysis of time-oriented clinical data, modeling and execution of guidelines, natural language generation of clinical data.

Introduction to Information Security*		**3 credits**
Prerequisites:	**Linear Algebra A + B, Operating Systems, Computer Communications, Computer Architecture** **Introduction to Encryption must be taken at the same time**	**MSc Semester 2**
Lecture:	**Mr. Binyamin Hirshberg**	**3 weekly hours**

Information security objectives, general threats to information systems. The basics of encryption theory, cryptography engineering: symmetric and asymmetric ciphers (DES, AES), block and stream ciphers (RC4). Key exchange methods (Diffie-Helman). HASH algorithms for cryptographers and digital signatures. Public key infrastructure certificates (PKI). Security policies and protection fundamentals: depth protection, separation of liability and authority, identification, verification, permissions and minimum right principle, Bell LaPadua model. Concepts and techniques of attacks. Attackers and ethics. Network security basics, SSL case analysis, computer security basics, operating systems and trusted computing. Android case analysis.

Introduction to Encryption		**3 credits**
Prerequisites:	**Algebraic Structures, Algorithms**	**Year 3 – Semester 2**
Lecture:	**Dr. Laure Barthel**	**3 weekly hours**

How do I send a confidential message? How do I sign a message on my computer? How do you share secrets? How do you prove identity? How do you flip a coin online? In the course we will see some answers. Topics covered: Classic encryption systems: block cipher, stream cipher, introduction to DES. Public encryption systems: introduction, RSA algorithm, RSA attacks, discrete log, digital signatures, hash functions. Elliptic curves and their uses in cryptography. Zero knowledge proofs. Secret sharing. During the semester, we will cover the mathematical concepts needed for encryption such as the integers modulo n and finite fields.

Artificial Intelligence (will be taught in 2020 only on Women's Studies)		**3 credits**
Prerequisites:	**Discrete Mathematics, Introduction to Theoretical Computer Science**	**BSc – Semester 2**
Lecture:	**Dr. Esther David**	**3 weekly hours**

Introduction to Artificial Intelligence. Troubleshooting. Search methods (non-informational search, informational search, heuristic search, local search, genetic algorithms). Search under rivalry conditions – games. Constraint satisfaction issues. Representation and knowledge acquisition through logic. Classic design. Game theory. Learning from examples. Neuronal networks.

Seminar in Computational Geometry		**2 credits**
Prerequisites:	**Algorithms**	**Year 3 – Semester 1**
Lecture:	**Dr. Laure Barthel**	**2 weekly hours**

What is the connection between a giraffe and your nearest post office? How do you program a robot's movement? How does the computer know which image you clicked on? How does GPS find the neighborhood map? You will learn about all these problems and many others in the course. With the help of these examples, we will introduce some of the techniques and data structures of computational geometry (for example, sweeping line algorithms, DCEL). Students will prepare a lecture and written work.

Big Data Seminar		2 credits
Prerequisites:		
Lecture:	**Dr. Ayelet Goldstein**	**2 weekly hours**

The amount of data that is currently accumulating in computer systems is growing at a dizzying pace, to the extent that traditional data processing applications can no longer cope with them. This reality requires the ability to process and analyze vast amounts of information in order to draw conclusions.
In the course, we will take a closer look at fundamental topics in the field of information. We will study current approaches and algorithms used in the field such as: google web search, recommendation systems, parallel computing, communities in social networks, clustering in big data, computational advertising and more.

Seminar on Communications & Distributed Systems		2 credits
Prerequisites:	**System Programming & Introduction to Parallel Programming, Scripting Operating Systems & Programming, Probability Theory, Algorithms**	**Year 3 – Semester 2**
Lecture:	**Dr. Martin Land**	**2 weekly hours**

The seminar will cover advanced topics in modern computer networks. Among the topics we will cover: protocols for organizing and managing networks, cloud networks, routing and switching integration, high-speed infrastructure (DSL, optic fibers and more), information security, wireless networks and cellular systems, performance analysis, and other topics in accordance with the participants' interest.

MSc Studies

MSc – Mandatory Studies

Courses marked with * will not be offered in 5780 (2020)

Protocols & Computer Networks		**3 credits**
Prerequisites:	**Algorithms, Operation Systems, Computer Networking**	**MSc – Semester 1**
Lecture:	**Dr. Martin Land**	**3 weekly hours**

Understanding the goals, technologies, algorithms, protocols and systems of computer communication and internetworking, with an emphasis on reading academic literature and standards documents and understanding methods for performance analysis and network design. Reading material for the course will be drawn from "classic" and innovative research articles and RFCs. Through submitting simulation exercises, students will become familiar with tools for researching networks and understanding their operation.

Object Oriented Analysis and Design	**Course code: 10251041**	**3 credits**
Prerequisites:	**Object Oriented Programming**	**MSc – Semester 1**
Lecture:	**Dr. Solange Karsenty**	**3 weekly hours**

This course aims to provide knowledge, in-depth understanding, and techniques for analyzing, designing, and building complex object-oriented software systems. The course includes study and practice of object oriented design principles, design patterns, design by contract, methods for testing and validating software systems, and aspect oriented programming.

Computational Complexity		**3 credits**
Prerequisites:	**Automata and Formal Languages, Computability & Computational Complexity**	**MSc – Semester 2**
Lecture:	**Dr. Asaf Nusbaum**	**3 weekly hours**

The aim of the course is to understand the basic issues and main results of the theory of computation, with an emphasis on sorting computational assignments into those that can be or cannot be effectively implemented. Main topics of the course: space complexity, Polynomial hierarchy, Cyclomatic (NC) complexity, random calculations, interactive proofs, PCP theorem and average-case complexity.

Final Project	**Course code: 10251271**	**10 credits**
Prerequisites:	**Completion of courses on MSc track**	**MSc – Semester 1/2**
Lecture:	**Dr. Solange Karsenty** **Prof. Michel Bercovier** **Dr. Yoram Yekutieli** **Prof. Michael Berman** **Dr. Martin Land**	**20 weekly hours**

Students will define topics for research projects under the academic supervision of faculty members. Project topics can be theoretical or applied. Students may also take advantage of additional supervision by external

researchers (academia or industry). The schedule for each project will be jointly built by the student and the academic supervisor.

Advanced Algorithms*	**Course code: 10251051**	**3 credits**
Prerequisites:	**Algebraic Structures, Algorithms**	**MSc – Semester 2**
Lecture:	**Dr. Arie Yakir**	**3 weekly hours**

The fundamental problem of algebra: rings, Euclidean rings, quotient rings. Arithmetic: discrete Fourier transform, polynomial multiplication, integer multiplication, matrix multiplication, the Chinese remainder theorem in a Euclidean ring, partial fraction decomposition. Finite fields: construction of finite fields, polynomial factoring over finite fields (linear algebra-based algorithms), construction of irreducible polynomials over finite fields. Modern methods for factoring in the ring of polynomials over the integers: factoring modulo a “large” prime number, factoring modulo a “small” prime number and raising to factoring modulo a power of the prime. Short vectors in lattices. Computational algebraic geometry: polynomials and affine varieties, monomial order, division with remainder in the ring of polynomials with several variables. Hilbert’s basis theorem and Grobner bases, Buchberger’s algorithm, applications in geometry.

Software Engineering*	**Course code: 10251041**	**3 credits**
Prerequisites:		**MSc – Semester 1**
Lecture:	**Mr. Yigal Cohen**	**3 weekly hours**

Understanding the application of software engineering in software projects. Establishment of a project work framework. Implementation of a software project from planning to completion, in practice. In-depth analysis of the various processes and development stages and the relationships between them. Details of the actions that support the project (measurements, risk management, quality management and more). Analysis and comparison of different classical and modern management practices while understanding the advantages and disadvantages of each and their amalgamation.

Advanced Computer Architecture*	**Course code: 10251031**	**3 credits**
Prerequisites:	**Computer Architecture**	**MSc – Semester 1**
Lecture:	**Dr. Martin Land**	**3 weekly hours**

Architecture is the field of computer science that provides a platform for the practical realization of innovations in programming and algorithms. After a brief summary of topics from the undergraduate course in computer architecture, the course introduces up-to-date performance enhancement methods: superscalar processors and instruction level parallelism, branch prediction behavior prediction, trace caching, multi-core processors and thread level parallelism, parallel programming support.

MSc – Elective Studies

Courses marked with * will not be offered in 5780 (2020)
Machine Learning Software, AI in Medicine – see BSc program

Man-Machine Interaction Seminar	**Course code: 10251291**	**2 credits**
Prerequisites:		**MSc – Semester 1**
Lecture:	**Dr. Solange Karsenty**	**2 weekly hours**

We will study and discuss man machine interaction in for traditional interactive systems, mobile systems and web applications. The seminar includes a collection of topics from current research trends: user centered design, usability, the world of Internet and users, systems and society, behaviors, security, mobile systems.

Cognitive Computing: Computer architectures and brain-inspired programming		**3 credits**
Prerequisites:	**Introduction to Object Oriented Programming & Software Engineering, Computer Architecture**	**MSc – Semester 1**
Lecture:	**Dr. Elishay Ezra**	**3 weekly hours**

Computations based on von Neumann architecture have faced limitations in recent years. Consequently, artificial neuron networks are drawing increasing attention. At its core is the goal of building machines that exceed the computational ability of the brain, with certain aspects of cognition. Such a computational paradigm was developed in part by IBM. IBM has developed the SyNAPSE chip, built from an unprecedented number of 1,000,000 neurons and 256 million synapses. It is the largest chip developed by IBM – it is built from 5.4 billion transistors that make up 4,096 neuro-synaptic cores and needs 70mW (sizing less than traditional chips). As part of a comprehensive ecosystem that integrates cognitive hardware and software, the technology breaks the boundaries of distributed computing and supercomputer development. In this course we will learn the basics of neuromorphic computing, focusing on analog and digital models for neuro-synaptic cores, new software paradigms for cognitive computations, and algorithms and applications for synaptic cores. The course does not require prior knowledge of electrical engineering or neurobiology, but students will need a considerable degree of thought flexibility and curiosity to delve a little into these topics.

Deep Learning		**3 credits**
Prerequisites:	**Linear Algebra A + B, Infinitesimal Calculus: Integral Uses & Approximate Calculations, Infinitesimal Calculus: Infinitesimal Calculus: Curves & Surfaces**	**MSc – Semester 2**
Lecture:	**Dr. Yoram Yekutieli**	**3 weekly hours**

Deep learning has burst into media attention in recent years as the main generator of the AI revolution, but in fact it is a pretty old concept. The idea and its realization are now booming because of three main factors: a vast amount of data, a great deal of computing power and learning algorithms that utilize the other two factors effectively.
In the course we will introduce the field of deep learning and its connection to other fields. We will use up-to-date tools to accomplish tasks that until recently were considered science fiction. We will learn about limitations of the tools and methods and describe future development directions.
We will focus on the practical aspects of deep learning:
Collecting existing data and databases, selecting models, using existing models and transfer learning, using libraries and software frameworks (such as TensorFlow, Theano, Keras, Matlab Neural Network Toolbox),

troubleshooting model training, cloud computing, deploying and using models on different end devices and under different conditions.
Exercises and demonstrations show actual use of deep learning to solve a variety of problems such as: text and natural language processing, recommendation systems, style text production, image processing, image search, style recognition and application in pictures, image production with autoencoders, handling audio and music, game resolution and others problems with deep reinforcement learning.

Euclidean geometry, Curves & Polyhedrons		**3 credits**
Prerequisites:	**Linear Algebra A + B, Algebraic Structures**	**MSc – Semester 2**
Lecture:	**Dr Arie Yakir**	**3 weekly hours**

Affine geometry: flats, affine transformation, the structure of the general affine group. Euclidean geometry: Euclidean vector space, the general orthogonal group, angle concepts, affine Euclidian space, the group of isometries, characterization of the isometry group in 2 dimensional and in 3 dimensional space, the geometry of the triangle, spheres, convexity, Euler's formula and convex polytopes.

Information Security		**3 credits**
Prerequisites:	**Linear Algebra A + B, Operation Systems, Computer Communications, Computer Architecture**	**MSc – Semester 2**
Lecture:	**Mr. Binyamin Hirshberg**	**3 weekly hours**

Information security objectives, general threats to information systems. The basics of encryption theory, cryptography engineering: symmetric and asymmetric ciphers (DES, AES), block and stream ciphers (RC4). Key exchange methods (Diffie-Helman). HASH algorithms for cryptographers and digital signatures. Public key infrastructure certificates (PKI). Security policies and protection fundamentals: depth protection, separation of liability and authority, identification, verification, permissions and minimum right principle, Bell LaPadua model. Concepts and techniques of attacks. Attackers and ethics. Network security basics, SSL case analysis, computer security basics, operating systems and trusted computing. Android case analysis.

Mobile Robots*		**3 credits**
Prerequisites:	**Algebra A + B, Infinitesimal Calculus: Integral Uses & Approximate Calculations, Infinitesimal Calculus: Infinitesimal Calculus: Curves & Surfaces**	**MSc – Semester 2**
Lecture:	**Dr. Yoram Yekutieli**	**3 weekly hours**

Automated airplanes, automated cars, automated vacuum cleaners and anthropomorphic robots (such as Sony from the movie I Robot) are all mobile robots. These sophisticated devices (and other creatures moving around the world) face many complex challenges, including:

- Orientation (Where am I? How do I map my surroundings? How do I navigate to different places? How will I plane a route? How will I avoid obstacles?)
- Use of sensors (What can I see, hear, feel? What is the distance to objects? Are they moving?)
- Communications (How should I communicate with my peers/other creatures? How do I produce sound / light / radio effectively? Is communication open or confidential?)
- Action on the world (How do I move in different areas? How do I manipulate objects?)
- Use of limited resources (How can I preserve energy / power? How can I be effective in my computational-cognitive resource use?
- Planning, strategy, learning and understanding the world (How will I plan my actions according to the tasks and information I gather from the world? What is best to do in uncertain and changing conditions? How will other opponents react? How will I improve my skills and performance?)

These problems include aspects of mechanics and hardware, software and information. In the course we will describe these problems and some of the solutions and will experiment with construction, programming and operation of LEGO NXT mobile robots.

Scientific Calculation Methods*	**Course code: 10251081**	**3 credits**
Prerequisites:	**Linear Algebra A + B, Analysis A + B**	**Year 1 – MSc – Semester 2**
Lecture:	**Dr. Yoram Yekutieli**	**3 weekly hours**

Discrete and continuous models, dynamic systems with one and two variables, phase plane analysis and finding fixed points, linearization and stability analysis with eigenvalues, nullclines, limit cycles, Hopf bifurcation, bistability, examples from the natural sciences. Differential equations and numerical solutions, characterization of methods according to efficiency and stability, adaptive step size, uses of Lagrange multipliers to solve dynamics with constraints and numerical solutions. Diffusion limited aggregation systems, cellular automata and use of the maze solving algorithm. Markov chains, absorption, ergodic chains and state distribution. Linear interpolation approximations and solving equations. Search problems in large spaces, optimization, uphill (downhill) climb methods, escape from local maxima and minima, genetic algorithms and evolutionary computing. Dimensionality reduction, principal component analysis, separation of sources by independent component analysis. Classification and clustering K-means, expectation maximization, maximum likelihood estimation, geometric methods. Correlation analysis, frequency space and information theory.
Medical robotics: robots for surgery, laparoscope and endoscope manipulators, micro robots, remote operation, team training simulators, machine interfaces.

Computer Vision & Advanced Learning*	**Course code: 10251071**	**3 credits**
Prerequisites:	**Linear Algebra A + B, Analysis A + B**	**MSc – Semester 1**
Lecture:	**Dr. Yoram Yekutieli**	**3 weekly hours**

Can a computer see? In the past, this question was discussed mainly in fiction, but today the applications that use computer vision are increasing. Process control in factories, detecting lane departure, tracking suspects at airports and identifying faces are examples of commercial use of computer vision systems.
In this course, we will introduce the field and systematically describe the geometry of computer vision systems. The subjects we will study are: camera parameters, perspective projection, and affine projection. Projective geometry and its uses. Rotation and translation transformations in 2D and 3D, homogeneous coordinates, coordinate system transformation. Projection matrices. Camera calibration: solving systems of equation using least square error method, solution $Ax = 0$, $Ax = b$, pseudo-inverse. Evaluation of the projection matrix using linear and nonlinear methods. Internal and external parameters of the camera, decomposition the projection matrix. Radial deformations. Automatic calibration. The geometry of several views: epipolar geometry, calculating the fundamental matrix directly and using matching points. Epipolar image rectification. Three views. Stereoscopy and 3D reconstruction. Use of correlations to perform matching. Dense reconstruction or according to selected points. Radiometry – light measurement: light in space and on surfaces. Shading light sources and structure identification. Use of RANSAC, Hough transform, Harris corner detection for robust estimation. Dimensionality reduction and face recognition with PCA. Using a lot of information (Internet) for identification: detection and recognition.

Robotics*		**Course code: 10251111**	**3 credits**
Prerequisites:	**Linear Algebra A + B, Analysis A + B**		**MSc – Semester 2**
Lecture:	**Dr. Yoram Yekutieli**		**3 weekly hours**

Basic kinematics: transformation of coordinate systems. Denavit-Hartenberg (DH) parameters, Modified DH. Degrees of freedom and kinematics of a robotic manipulator. Forward and inverse kinematics: workspace, multiple solutions, solution methods. Differential motion: linear and angular velocity, Jacobian method, Jacobian calculation. Path planning: Cartesian space and joint angle space, path planning methods. Basic statistics and dynamics: the relationship between statics, kinematics and Jacobian. Movement, trajectory and navigation planning: configuration space, main planning methods.

Geometry*		**3 credits**
Prerequisites:	**Linear Algebra A + B, Algebraic Structures**	**MSc – Semester 2**
Lecture:	**Dr. Arie Yakir**	**3 weekly hours**

Affine geometry: directness, affine transformation, general affine set structure, Thales's theorem, Menelaus's theorem, Ceva's theorem, Pappus's theorem, Desargues theorem. Projective geometry: subspaces, affine space projective completion, Pappus and Desargues projective versions, The general projective set, topology. Euclidean geometry: Euclidean vector space, the general orthogonal set, angle concepts, affine Euclidian space, the isometric set, characterization of the isometric set in 2D and 3D space, triangle geometry, spheres, curves and curved polyhedrons. Non-Euclidean geometry: axiomatic approach to geometry, elliptical geometry, hyperbolic geometry.

Error Correction Codes*		**3 credits**
Prerequisites:	**Linear Algebra A, Linear Algebra B, Discrete Mathematics, Algebraic Structures**	**MSc – Semester 1**
Lecture:	**Dr. Arie Yakir**	**3 weekly hours**

Basic concepts: linear ciphers, generator matrix and test matrix, distance and weight, coding equivalents, encryption and decryption, the syndrome method. Bounds: sphere-packing bound, Plotkin bound, Johnson bound, MDS codes, sphere-packing bound. Finite fields: polynomials and the Euclid's algorithm, primitive root modulo, finite field construction, automorphisms, Galois extension. Cyclic cyphers: BCH code, BCH code algorithms.

User Interface Software Technology*		**Course code: 10251111**	**3 credits**
Prerequisites:	**Object Oriented Programming**		**MSc – Semester 1**
Lecture:	**Dr. Solange Karsenty**		**3 weekly hours**

This course covers man machine interfaces software tools for interactive applications, advanced toolkits and visual programming, graphic user interface builders, as well as basic principles for designing and building user-friendly interfaces.

Topics of the course include: conceptual model, task analysis, and user-centered design software engineering concepts, model view controller architecture, components, widgets, input techniques, event-driven programming, windows systems, dialog flow, error management and undo , screen design layout and graphic design principles for web programming, modern development tools and frameworks.

User Interface Design*		**3 credits**
Prerequisites:	**User Interface Software Technology**	**MSc – Semester 2**
Lecture:	**Dr. Solange Karsenty**	**3 weekly hours**

The aim of the course is to study design processes and build advanced human-machine interfaces. Topics: Human factors, the human processor and conceptual models, task analysis, user-centered design, usability and evaluation, graphic design and means of display, virtual reality, augmented reality, web design.