Seconda Parte: Programmi e docenti degli Insegnamenti per l'A.A. 2018/19

In questa seconda parte della guida vengono riportati tutti gli insegnamenti attivati nell'anno accademico 18/19 presso il Corso di Laurea di Informatica, di interesse per gli studenti e le studentesse immatricolati nell'anno e per gli studenti e le studentesse delle due coorti precedenti che devono seguire gli insegnamenti del secondo e terzo anno rispettivamente. Sono inoltre riportati gli insegnamenti che verranno attivati nei prossimi anni, nonché l'elenco dei codici per le convalide (per gli studenti e le studentesse in trasferimento).

Insegnamenti attivi nel 18/19.

La tabella che segue riporta tutti gli insegnamenti che saranno attivi nell'anno accademico 2018/19. Nella tabella le colonne "Codice", "Insegnamento" e "SSD" indicano, rispettivamente, il codice di Ateneo, il titolo dell'insegnamento e il suo settore scientifico disciplinare, TAF e Ambito individuano la tipologia dell'attività formativa (di base, caratterizzante, a scelta dello studente/della studentessa, etc.), "CFU" sono i crediti formativi associati all'insegnamento, "docenti" sono i docenti responsabili dell'insegnamento e "semestre" il periodo didattico in cui si tiene l'insegnamento. Tutti gli insegnamenti del primo e del secondo anno sono sdoppiati (corso A e corso B)*, mentre i laboratori sono quadruplicati (turni A1, A2, B1 e B2 oppure T1, T2, T3, T4. Ricordiamo che analoghe informazioni sono reperibili sulle pagine web del corso di laurea (da http://laurea.educ.di.unito.it/ seguire il link 'insegnamenti', nella sezione "Per chi studia con noi").

** Corso A: dalla A alla K, Corso B: dalla L alla Z*

Codice principale	INSEGNAMENTO	SSD	TAF	Ambito	CFU	DOCENTI	Semestre	Note
mfn0597	Algoritmi e Strutture Dati	inf/01	B	caratterizzante	9	**Horvath (Teoria A), de' Liguoro (Teoria B), Magro (lab T1), Esposito (lab T2), Pozzato (lab T3), Pozzato (lab T4), Damiani (compresenza sui 4 turni di laboratorio)**	2 sem	
mfn0570	Analisi Matematica	mat/05	A	di base	9	**A: Barutello, Seiler B: Dambrosio, Colasuonno**	2 sem	
mfn0586	Architettura degli Elaboratori	inf/01	A	di base	9	**Gaeta (teoria A e lab A1), Aldinucci (teoria B), Garetto (lab A2), Lucenteforte (lab B1), Schifanella C (lab B2)**	2 sem	
mfn0602	Basi di Dati	inf/01	B	caratterizzante	9	**Anselma (Teoria A) Anselma (lab T1), Pozzato (lab T2)Pensa (Teoria B), Da assegnare (lab T3 e T4)**	2 sem	
mfn0612 (inf0090 dal 19-20)	Calcolabilità e Complessità A (Calcolabilità e Complessità dal 19-20)	inf/01	D	a scelta	6	**Berardi**	1 sem	
mfn0588	Calcolo Matriciale e Ricerca Operativa	mat/09	A	di base	6	**Grosso (A), Aringhieri (B)**	1 sem	
mfn0604	Economia e Gestione dell'Impresa e Diritto	6 CFU secs-p/08 e 3 CFU ius/02	C	affine e integrative	9	**Pironti (Economia A e B), Montalcini/Sacchetto (Diritto A e B)**	2 sem	
mfn0617	Economia e	secs-	D	a scelta	6	**Pironti**	1 sem	

	Gestione dell'Innovazione	p/08						
mfn0600	Elementi di Probabilità e Statistica	mat/06	C	affine e integrative	6	**Sirovich Roberta (A e B)**	1 sem	
mfn0598	Fisica	fis/01	C	affine e integrative	6	**Migliore (A), Gagliardi (A), Pesando (B)**	2 sem	
mfn1353	Interazione Uomo Macchina	inf/01	D	a scelta	6	**Patti, Segnan**	1 sem	mutua su modulo di Interazione Uomo Macchina e Tecnologie Web
mfn0608	Interazione Uomo Macchina e Tecnologie Web	inf/01	B	caratterizzante	12	**Ardissono, Patti, Segnan**	1 sem	
mfn0590	Lingua Inglese I	L-Lin/12	E	conoscenza lingua straniera	3	**Griffin (esercitatore), Radicioni, Bini, Bono, Cordero, Patti (responsabili)**	1 e 2 sem	
mfn0610	Linguaggi e Paradigmi di Programmazione	inf/01	B	caratterizzante	9	**Bono, Cardone, Padovani**	1 sem	
mfn1354	Linguaggi e Paradigmi di Programmazione	inf/01	D	a scelta	6	**Bono, Cardone, Padovani**	2 sem	mutua su modulo di Linguaggi e Paradigmi di Programmazione da 9 CFU
mfn0603	Linguaggi Formali e Traduttori	inf/01	B	caratterizzante	9	**Zacchi (prof. a contratto Teoria A), Patti (lab T1), Sproston (lab T2), Coppo (Teoria B), Sproston (lab T3), Patti (lab T4)**	1 sem	
inf0003	Logica per l'Informatica	mat/01	D	a scelta	6	**Ronchi della Rocca prof. a contratto**	1 e 2 sem	
mfn0578	Matematica Discreta e logica	6 CFU mat/02 + 6 CFU mat/01	A	di base	12	**Ardizzoni (Mat Discr A), Viale (Log A) Mori (Mat Discr B), Motto Ros (Log B)**	1 sem	
mfn0633	Metodi Formali dell'Informatica	inf/01	B	caratterizzante	9	**de' Liguoro**	1 e 2 sem	
mfn0582	Programmazione I	inf/01	A	di base	9	**Cardone (teoria A), Gliozzi (lab A1) Mazzei (lab A2), Roversi (Teoria B, lab B1, lab B2)**	1 sem	
mfn0585	Programmazione II	inf/01	A	di base	9	**Padovani (teoria A), Berardi (teoria B), Damiani (lab A1-A2), Torta (lab B1), Magro (lab B2)**	2 sem	
mfn0605	Programmazione III	inf/01	B	caratterizzante	6	**Ardissono**	1 sem	
mfn0635	Reti di Elaboratori	inf/01	B	caratterizzante	12	**Garetto, Sereno**	1 e 2 sem	
mfn1362	Reti I	inf/01	B	caratterizzante	6	**Botta**	1 sem	
inf0002	Servizi Web	inf/01	D	a scelta	6	**Ardissono**	1 sem	mutua su modulo di Interazione Uomo Macchina e Tecnologie Web
mfn0636	Sicurezza	inf/01	B	caratterizzante	6	**Bergadano**	2 sem	
mfn0618	Sistemi Informativi	inf/01	D	a scelta	6	**Micalizio**	1	
mfn0607	Sistemi Intelligenti	inf/01	B	caratterizzante	6	**Baroglio**	2 sem	
mfn0601	Sistemi Operativi	inf/01	B	caratterizzante	12	**Gunetti (Teoria A), Baroglio (Teoria B), De Pierro (lab T1 C), Radicioni**	1 sem	

						(lab T1 Unix), Bini (lab T2 C e T2 Unix; lab T3 C e T3 Unix), De Pierro (lab T4 C), Schifanella C (lab T4 Unix)		
mfn0606	Sviluppo delle Applicazioni Software	inf/01	B	caratterizzante	9	**Baldoni (Teoria), Capecchi (lab 1), Picardi (lab 2)**	2 sem	
inf0004	Storia dell'Informatica	inf/01	D	a scelta	6	**Gunetti/Cardone**	1 e 2 sem	
mfn0634	Tecnologie Web	inf/01	B	caratterizzante	6	**Ruffo**	1 sem	
Inf0073	Stage		F	altre attività	9			
Inf0072	Prolungamento stage		D	a scelta	6			
Inf0074	Prova Finale		E	altre attività	3			

Codici per convalide di insegnamenti e competenze.

Gli studenti e le studentesse che per effetto di passaggio o trasferimento abbiano ottenuto la convalida di crediti liberi con nessuna corrispondenza specifica a insegnamenti del Corso di Laurea in Informatica, sono pregati di selezionare i codici contenitore MFN1522, MFN1409 e/o MFN1408 in base alla delibera della Commissione Passaggi e Trasferimenti, integrando eventualmente con altro insegnamento libero da selezionare dalla lista prevista per il percorso scelto, in caso non sia ancora stato raggiunto il minimo di 12 CFU.

mfn1409	Altre Attività	altre attività, a scelta lettera A	12
mfn1408	Altre Attività	altre attività, a scelta lettera A	6
mfn1522	Altre Attività	altre attività, a scelta lettera A	6

Informazioni aggiornate al 1 ottobre 2018

Programmi e altre informazioni per gli insegnamenti attivi nel 18/19 (syllabus degli insegnamenti)

Questa parte sarà disponibile a settembre, come da regolamento didattico di Ateneo. Gli studenti e le studentesse possono intanto consultare le pagine degli insegnamenti del 18/19 sul sito web del corso di studio, che sono in fase di ultimazione in questi giorni.

Insegnamento **MFN0597 - Algoritmi e Strutture Dati**

Insegnamento (inglese):	
CFU:	**9**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Ferruccio DAMIANI (Co-presenza in aula)** **Ugo DE' LIGUORO (Titolare)** **Roberto ESPOSITO (Titolare)** **Andras HORVATH (Titolare)** **Diego MAGRO (Titolare)** **Gian Luca POZZATO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Si presuppone che lo studente sia a conoscenza delle basi della programmazione e dei linguaggi di programmazione Java e C; che sia in possesso delle nozioni elementari della matematica del discreto, del continuo, e della logica matematica.

Eventuali corsi propedeutici

Programmazione I & Laboratorio, Programmazione II & Laboratorio, Matematica Discreta, Analisi Matematica, Logica Matematica, Sistemi Operativi (in particolare l'apprendimento del linguaggio C).

2. Obiettivi formativi:

L’insegnamento ha lo scopo di introdurre i concetti e le tecniche fondamentali per l’analisi e la progettazione di algoritmi, che sono alla base dello sviluppo del software. Gli studenti acquisiranno conoscenze circa l’analisi di correttezza e complessità computazionale degli algoritmi, sulle strutture dati per la rappresentazione dell’informazione, sulle tecniche di problem solving mediante lo sviluppo di algoritmi efficienti. L’insegnamento è supportato da un laboratorio che ne costituisce parte integrante, finalizzato alla realizzazione e sperimentazione degli algoritmi e delle strutture dati mediante un linguaggio imperativo ed uno object-oriented.

3. Risultati dell'apprendimento attesi:

Al termine del corso ci si aspetta che lo studente sia in grado di risolvere problemi algoritmici elementari mediante la progettazione degli algoritmi e l’analisi delle soluzioni adottate, sia per quanto riguarda l’efficienza, sia per quanto concerne la scelta delle strutture dati manipolate dagli algoritmi. Lo studente sarà in grado di individuare e comprendere algoritmi illustrati nella letteratura specialistica e di adattarli realizzando concretamente le proprie implementazioni, tanto producendo il codice dei programmi quanto utilizzando librerie software preesistenti. Infine lo studente dovrà essere in grado di comunicare e documentare le soluzioni adottate in modo tecnicamente rigoroso.

4. Modalità di verifica dell'apprendimento:

L'esame di Algoritmi e strutture dati consiste in una prova scritta ed in una discussione orale del progetto di laboratorio. Il superamento dello scritto permette di accedere alle prove orali della sessione in cui è stato superato lo scritto. Nel caso in cui questa seconda prova non venga superata entro i termini della sessione, lo scritto dovrà essere ripetuto.

Il voto sarà la media pesata dei voti ottenuti nelle due prove scritta ed orale, valutate in 30+1

esimi, essendo comunque necessario il raggiungimento della sufficienza in entrambe le prove.

5. Modalità d'insegnamento:

Sono previste 60 ore di lezione frontale ed esercitazioni in aula, e 30 ore di attività guidata dai docenti in laboratorio. Per le indicazioni bibliografiche, la distribuzione di note ed altro materiale didattico, e per la consegna e valutazione degli elaborati delle esercitazioni si farà uso della piattaforma e-learning Moodle, cui si richiede agli studenti l'iscrizione.

6. Attività di supporto:

Il materiale didattico sarà reso via via disponibile on line attraverso la piattaforma Moodle.

7. Programma:

Programma

- Problemi e algoritmi: risolubilità, correttezza, complessità.
- Analisi computazionale e complessità asintotica
- Algoritmi di ordinamento
    - Algoritmi elementari quadratici
    - Divide et impera: mergesort e quicksort
    - Risoluzione di relazioni di ricorrenza
    - Limiti inferiori per l'ordinamento
- Programmazione dinamica
    - Massima sottosequenza comune
- Strutture dati
    - Strutture concrete: array, liste, tabelle hash
    - Strutture astratte: pile, code, dizionari
    - Code di priorità, heapsort
    - Analisi ammortizzata, cenni
- Alberi
    - Definizione e visita
    - Alberi di ricerca
    - Alberi rosso-neri
- Grafi
    - Definizione e visita
    - Ordinamento topologico e componenti fortemente connesse
    - Algoritmi greedy: alberi di copertura minima
    - Cammini minimi: algoritmo di Dijkstra

8. Testi consigliati e bibliografia:

1. Per la teoria:
    - Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: Introduzione agli algoritmi e strutture dati, McGraw-Hill 2010.
2. Per il laboratorio:
    - Horstmann: Concetti di informatica e fondamenti di Java, Quinta Edizione (o successiva), Apogeo, 2010.

Insegnamento	**MFN0570 - Analisi Matematica**

Insegnamento (inglese):	**Mathematical Analyis**
CFU:	**9**
Settore:	**MAT/05 - ANALISI MATEMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**A - di base**
Docenti:	**Vivina Laura BARUTELLO (Titolare)** **Walter DAMBROSIO (Titolare)** **Joerg SEILER (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

L’insegnamento prevede la conoscenza dei contenuti di matematica di base forniti dalla scuola secondaria di secondo grado.

In particolare, a livello di conoscenze e comprensione in ingresso lo studente dovrà:

â□□ conoscere i concetti di base sulla retta, sia dal punto di vista della geometria sintetica sia della geometria analitica, con particolare riferimento al concetto di pendenza;

â□□ conoscere le funzioni quadratiche e le loro proprietà algebriche e grafiche;

â□□ ricordare le proprietà delle potenze e dei logaritmi e conoscere i grafici delle funzioni potenza, esponenziali e logaritmiche;

â□□ conoscere gli elementi essenziali di trigonometria (misure degli angoli in radianti, grafici delle funzioni circolari);

â□□ conoscere i concetti di dominio, immagine, zeri, segno e monotonia per funzioni reali di una variabile reale.

Inoltre, come applicazione di conoscenza e comprensione, lo studente dovrà saper:

â□□ determinare l’equazione della retta passante per un punto ed avente pendenza assegnata e l’equazione della retta passante per due punti;

â□□ determinare l’equazione di una retta a partire dal suo grafico (calcolo di pendenza e intercetta);

â□□ tracciare il grafico di una funzione lineare e determinare per via grafica dominio, immagine, zeri e segno, monotonia;

â□□ tracciare il grafico di una funzione quadratica e determinare per via grafica dominio, immagine, zeri e segno, monotonia;

â□□ riconoscere come varia la retta tangente al grafico di una funzione quadratica in un suo punto, anche in relazione alla concavità della funzione;

â□□ risolvere equazioni e disequazioni di primo e secondo grado, anche per via grafica;

â□□ tracciare il grafico di funzioni potenza xa, con a intero positivo o negativo, e determinarne per via grafica dominio, immagine, simmetrie, zeri e segno, monotonia, massimi e minimi;

â□□ tracciare il grafico di funzioni del tipo a^x o log_a x, con a positivo, e determinarne per via grafica dominio, immagine, zeri e segno, monotonia;

â□□ risolvere equazioni e disequazioni del tipo a^x=b, a^x>b, a^x <b, log_a x=b, log_a x>b e log_a x<b;

â□□ trasformare la misura di un angolo da gradi a radianti e viceversa;

â□□ tracciare il grafico delle funzioni circolari;

â□□ risolvere equazioni del tipo sin x =b e cos x=b;

â□□ determinare dominio, immagine, zeri e segno, monotonia di una funzione a partire dal suo grafico.

Eventuali corsi propedeutici

I prerequisiti richiesti potranno essere recuperati attraverso il Corso di Riallineamento di Matematica presente sulla Piattaforma Orient@mente.

2. Obiettivi formativi:

L'insegnamento ha lo scopo di presentare le nozioni di base su funzioni, grafici e loro trasformazioni, di introdurre i concetti di derivata e di integrale definito e di illustrare l'utilizzo di tecniche di tipo analitico nello studio di fenomeni discreti. Si tratta di argomenti indispensabili per la formazione dei laureati in scienze dell'informazione (classe L-31).

L'insegnamento concorre agli obiettivi formativi dell'area Matematico-Fisica del corso di Laurea in Informatica, fornendo conoscenze relative ai concetti ed agli strumenti matematici e metodologici fondamentali necessari per descrivere, schematizzare e interpretare i principali aspetti della realtà che ci circonda. In particolare, l'insegnamento si inoltre propone di accrescere le capacità di comprensione degli studenti e di consentire loro di acquisire un modo rigoroso ed analitico di ragionare e affrontare nuovi problemi.

La significativa presenza di teoremi, molti dei quali con dimostrazione, ha l'obiettivo di rafforzare nello studente le attitudini logico-deduttive apprese nel corso di Matematica discreta e Logica.

3. Risultati dell'apprendimento attesi:

Conoscenza e comprensione

Alla fine di questo insegnamento lo studente saprà:

â□□ riconoscere i grafici e le proprietà asintotiche delle funzioni elementari;

â□□ rievocare e ricordare la definizione di funzione monotona e spiegare e descrivere il legame tra la monotonia di una funzione composta e quella delle funzioni componenti;

â□□ spiegare e illustrare il significato intuitivo di limite, anche in riferimento alla sua interpretazione in termini grafici;

â□□ riconoscere la definizione di limite, sia nel caso continuo (funzioni), sia nel caso discreto (successioni);

â□□ ricordare e rievocare la definizione ed il significato di asintoto verticale o orizzontale;

â□□ enunciare e dimostrare alcuni risultati fondamentali sui limiti, illustrandone il significato anche a livello grafico;

â□□ riconoscere e ricordare la definizione di funzione continua ed illustrare criticamente il suo legame con il concetto di limite;

â□□ enunciare i principali risultati relativi alle funzioni continue;

â confrontare le diverse crescite di una funzione all’infinito e distinguere tra crescita logaritmica, polinomiale ed esponenziale, sia in termini grafici sia in termini quantitativi;

â ricordare le definizioni dei simboli di Landau e spiegare le relazioni che intercorrono tra i vari simboli;

â ricordare la definizione di successione geometrica, in termini ricorsivi o iterativi, e illustrarne le principali applicazioni;

â discutere il comportamento asintotico e le proprietà qualitative di una successione geometrica in dipendenza dai valori della sua ragione;

â riconoscere una successione definita per ricorrenza;

â discutere la stabilità dell’equilibrio di una successione per ricorrenza lineare del primo ordine;

â spiegare l’utilizzo dei corretti simboli di Landau nella discussione della complessità di un algoritmo e determinare tale complessità in alcuni semplici casi;

â riconoscere e ricordare la definizione di derivata di una funzione in un punto;

â interpretare il concetto di derivata di una funzione in un punto da vari punti di vista applicativi (pendenza, velocità istantanea, tasso istantaneo di variazione);

â ricordare l’espressione della retta tangente al grafico di una funzione in un punto;

â illustrare il legame tra funzioni continue e funzioni derivabili, fornendo ove necessario dimostrazioni, esempi e controesempi;

â rievocare e ricordare l’espressione della derivata delle funzioni elementari e le regole di derivazione;

â ricordare la definizione di primitiva di una funzione ed illustrare il legame tra diverse primitive della stessa funzione su un intervallo;

â enunciare il Teorema di Lagrange ed illustrarne le sue interpretazioni cinematica e geometrica e le sue principali conseguenze (legame tra la monotonia di una funzione ed il segno della sua derivata, caratterizzazione delle funzioni a derivata ovunque nulla su un intervallo, legame tra diverse primitive della stessa funzione su un intervallo);

â collegare le proprietà geometriche di concavità/convessità del grafico di una funzione con le proprietà di segno e monotonia delle derivate della funzione;

â dimostrare i principali risultati relativi al calcolo differenziale;

â spiegare il problema dell’approssimazione locale di una funzione in un punto;

â ricordare e riconoscere l’espressione dei polinomi di Taylor di una funzione in un punto, valutando in termini anche quantitativi l’errore commesso approssimando una funzione con un suo polinomio di Taylor;

â ricordare i polinomi di Maclaurin delle funzioni elementari;

â illustrare il problema della risolubilità esatta ed approssimata di un’equazione;

â enunciare e dimostrare il Teorema di esistenza degli zeri;

â descrivere il metodo di Newton per la risoluzione approssimata di un’equazione, discutendone le proprietà di convergenza;

â enunciare e dimostrare il Teorema sulla convergenza del metodo di Newton per la risoluzione approssimata di un’equazione;

â confrontare il metodo di bisezione ed il metodo di Newton per la risoluzione approssimata di un’equazione, evidenziando gli aspetti vantaggiosi in ognuno di essi;

â descrivere la definizione di integrale definito di una funzione su un intervallo;

â interpretare la definizione di integrale definito in termini di lavoro, spostamento netto e valor medio;

â collegare il concetto di integrale definito con quello di area di regioni piane;

â ricordare la formula del punto medio per il calcolo approssimato di un integrale definito e valutare l’errore commesso nell’approssimazione;

â rievocare e spiegare le proprietà di linearità e monotonia dell’integrale definito;

â enunciare il Teorema fondamentale del calcolo integrale ed interpretarlo criticamente, evidenziando la sua centralità rispetto alle nozioni di derivata e integrale definito;

â enunciare il Teorema di Torricelli-Barrow e illustrare le sue conseguenze sul calcolo esatto di un integrale definito;

â dimostrare i principali risultati sul calcolo integrale;

â spiegare il concetto di integrale improprio, in relazione al problema dell’area di una regione illimitata di piano;

â enunciare e dimostrare i principali risultati relativi alla convergenza di un integrale improprio;

â dimostrare la convergenza dell’integrale improprio della funzione gaussiana e spiegarne la sua importanza in statistica;

â spiegare il concetto di serie ed illustrarne il legame con il concetto di successione;

â ricordare la definizione di serie convergente, divergente o indeterminata;

â ricordare la serie geometrica e discuterne la convergenza in dipendenza dalla ragione;

â discutere alcune applicazioni della serie geometrica;

â ricordare le serie armoniche generalizzate e discuterne convergenza e divergenza;

â enunciare e dimostrare i principali risultati teorici sulla convergenza di una serie;

â spiegare il legame tra integrali impropri e serie, enunciando e dimostrando il teorema che li mette in relazione, sotto opportune ipotesi.

Applicare conoscenza e comprensione

Alla fine di questo insegnamento lo studente avrà sviluppato capacità di lavorare sia su aspetti grafici sia su aspetti di calcolo, approssimato o esatto.

In particolare, a livello di grafici saprà:

â dedurre dal grafico di una funzione informazioni qualitative e quantitative sulla funzione stessa (dominio, immagine, monotonia, zeri, segno, limiti);

â ottenere dal grafico di una funzione il grafico di nuove funzioni, mediante trasformazioni geometriche o mediante l’uso delle proprietà delle funzioni composte;

â tracciare il grafico di funzioni ottenute da funzioni elementari mediante composizioni;

â stimare il valore della derivata della funzione in un punto a partire dal grafico di una funzione;

â□□ tracciare il grafico della derivata di una funzione a partire dal grafico della funzione stessa, analizzando in modo critico i legami tra una funzione e la sua derivata;

â□□ interpretare il grafico della numerosità di una popolazione in funzione del tempo e tracciare da esso il grafico del tasso di crescita in funzione del tempo;

â□□ tracciare il grafico della velocità di un oggetto che si muove di moto rettilineo, a partire dal grafico della sua posizione;

â□□ tracciare il grafico della posizione di un oggetto che si muove di moto rettilineo, a partire dal grafico della sua velocità e dalla posizione all'istante iniziale;

â□□ discutere la risolubilità di un'equazione, mediante il confronto tra grafici.

A livello di calcolo approssimato saprà:

â□□ determinare l'approssimazione di una funzione in un punto mediante un polinomio di Taylor e utilizzarla per stimare i valori della funzione;

â□□ stimare l'errore commesso nell'approssimazione locale di una funzione mediante un polinomio di Taylor;

â□□ eseguire ed implementare l'algoritmo di bisezione e lo schema ricorsivo del metodo di Newton per stimare le soluzioni di un'equazione;

â□□ calcolare in modo approssimato integrali definiti utilizzando la formula del punto medio, sia a partire dall'espressione esplicita della funzione integranda, sia a partire dal suo grafico;

â□□ stimare l'errore commesso nel calcolo approssimato di un integrale definito mediante la formula del punto medio.

A livello di calcolo esatto saprà:

â□□ determinare l'espressione di successioni per ricorrenza del primo ordine lineari;

â□□ calcolare la derivata di una funzione;

â□□ determinare l'approssimazione di una funzione in un punto mediante un polinomio di Taylor;

â□□ determinare la retta tangente al grafico di una funzione in un suo punto;

â□□ calcolare le primitive di funzioni, in casi immediati;

â□□ calcolare integrali definiti immediati;

â□□ determinare la somma di una serie geometrica, anche in funzione di eventuali parametri.

Inoltre, lo studente saprà:

â□□ discutere la convergenza di un integrale improprio o di una serie, utilizzando in modo opportuno i criteri di convergenza;

â□□ interpretare e rielaborare grafici qualitativi e dati quantitativi di fenomeni di tipo fisico (cinematica del punto) o biologico (popolazioni e loro dinamica).

Autonomia di giudizio

Alla fine di questo insegnamento lo studente saprà:

â□□ riconoscere ed individuare argomentazioni logiche con una chiara identificazione di assunti e conclusioni;

â□□ collegare e commentare criticamente i principali risultati teorici illustrati nel corso

dell’insegnamento, individuando i legami che tra essi intercorrono.

4. Modalità di verifica dell'apprendimento:

L'esame è strutturato in tre prove scritte, obbligatorie per tutti gli studenti, indipendentemente dall’Anno Accademico di iscrizione, tutte da sostenere nello stesso appello. Le prove si svolgono in successione: per accedere alla seconda è necessario aver superato la prima, per accedere alla terza aver superato la seconda.

Prima prova (quiz) La prima prova consiste nella risposta a cinque domande a scelta multipla che hanno l’obiettivo di verificare la capacità dello studente di risolvere semplici esercizi su argomenti di base. La durata è di venti minuti; per superare il test occorre rispondere in modo corretto ad almeno 4 domande su 5. L'esito è: superato o non superato ed è noto immediatamente al termine del test stesso; chi non supera il test non può accedere alla prova d'esame.

Seconda prova (teoria) La seconda prova verte sugli argomenti trattati a lezione ed esercitazioni e mira a valutare le conoscenze di definizioni e enunciati di teoremi, nonché le capacità logico-deduttive nell’esposizione delle dimostrazioni presentate a lezione. La prova è valutata in trentesimi; si accede alla terza prova con una valutazione almeno pari a 15/30.

Terza prova (esercizi) La terza prova prevede la risoluzione di un certo numero di esercizi, in cui lo studente deve dimostrare di saper applicare le sue conoscenze in problemi maggiormente strutturati. La prova è valutata in trentesimi ed è superata con una valutazione almeno pari a 18/30.

L’intero esame è superato se la media tra i risultati della seconda e terza prova è almeno pari a 18/30.

Modalità di iscrizione e regole per lo svolgimento degli esami La prenotazione agli appelli d'esame tramite Esse3 ed entro una settimana dalla data dell'esame è obbligatoria e indispensabile. Non si accetteranno prenotazioni pervenute via mail e non verranno ammessi studenti che non si siano prenotati. Inoltre, per una migliore organizzazione dei Laboratori informatici, chi si prenota e non si presenta all'esame senza prima avvisare i docenti non avrà diritto a partecipare all'appello successivo. Per sostenere l'esame è necessario presentarsi con un documento di riconoscimento (preferibilmente la smartcard) e ricordare le credenziali di Ateneo (username e password), che dovranno essere digitate sul computer dell'aula per iniziare le prove. Durante l'esame non è consentito l'uso di strumenti elettronici e non è permesso consultare testi o appunti. Durante la terza prova si può utilizzare la calcolatrice disponibile sul computer. E' assolutamente vietato, pena l'esclusione dall'esame, tenere alla postazione informatica telefoni cellulari, tablet e simili (anche se spenti, in tasca,..). La presenza di uno di questi apparecchi, anche spento, comporterà l'espulsione immediata dall'aula e l'annullamento della prova. Informazioni per gli studenti degli anni accademici passati

Gli studenti degli anni accademici passati devono sostenere l'esame secondo le modalità dell’anno accademico corrente.

5. Modalità d'insegnamento:

Le modalità di insegnamento comprendono: lezioni frontali, lezioni inverse (flipped), apprendimento attivo in aula e a distanza, esercitazioni in aula.

â□□ Lezioni frontali e attività in aula: lezioni frontali eventualmente supportate dall’uso di strumenti di videoscrittura e di software di visualizzazione dinamica; attività ed esercitazioni in aula con eventuale partecipazione degli studenti (svolgimento di esercizi, discussioni, gruppi di lavoro).

â□□ Attività e materiale online (Piattaforma Moodle): calendario delle lezioni e delle esercitazioni; video sostitutivi delle lezioni frontali per argomenti erogati in modalità inversa (flipped); quiz ed assegnazioni per l'apprendimento e l'autovalutazione.

L’insegnamento, con le sue modalità ed attività, contribuisce a formare e consolidare le seguenti competenze trasversali: gestione del tempo, attraverso lo svolgimento di prove di autovalutazione informatizzate aventi tempo stabilito; corretta attribuzione causale di successi ed insuccessi,

attraverso lo svolgimento di prove di autovalutazione con feedback da parte del tutor o dei docenti.

6. Attività di supporto:

Sono disponibili settimanalmente su piattaforma Moodle i Fogli di esercizi, in cui vengono proposti esercizi simili a quelli affrontati durante le esercitazioni. Questi esercizi possono essere svolti a casa oppure, in modo individuale o a piccoli gruppi, durante gli incontri settimanali del Tutorato disciplinare, in cui uno studente della LT in Matematica è a disposizione per aiutare nella loro risoluzione e per fornire chiarimenti e spiegazioni. Gli esercizi proposti svolgono la funzione di valutazione formativa in itinere ed hanno un ruolo assai importante nello studio degli argomenti del corso: il loro svolgimento e la loro correzione, effettuata negli incontri del tutorato, consentono allo studente di verificare costantemente la propria preparazione ed il grado di autonomia nel risolvere problemi simili a quelli affrontati durante le esercitazioni e della stessa difficoltà di quelli presenti all'esame.

Sono inoltre disponibili su piattaforma Moodle prove di autovalutazione e simulazioni delle prove d'esame.

7. Programma:

1. Funzioni, grafici e modelli

1.1. Funzioni elementari e loro grafici

1.2. Trasformazioni geometriche di grafici

1.3. Grafici di funzioni composte

2. Il concetto di limite

2.1. Il concetto di limite nel caso continuo (limite di funzioni)

2.2. Il concetto di limite nel caso discreto (limite di successioni)

2.3. Principali risultati teorici sui limiti

2.4. Successioni definite per ricorrenza

2.5. Crescite e confronti di crescite: i simboli di Landau

3. Calcolo differenziale

3.1. Derivata di una funzione in un punto

3.2. Funzione derivata e funzioni primitive; relazioni tra una funzione e la sua derivata o le sue primitive

3.3. Derivata e monotonia; derivata e convessità

3.4. Approssimazione locale di funzioni mediante polinomi

4. Risoluzione approssimata di equazioni

4.1. Il Teorema di esistenza degli zeri ed il metodo di bisezione

4.2. Il metodo di Newton

5. Calcolo integrale

5.1. Integrale definito di una funzione su un intervallo

5.2. Teorema Fondamentale del Calcolo Integrale

5.3. Teorema di Torricelli-Barrow

5.4. Integrali impropri

6. Serie numeriche

6.1. La serie geometrica

6.2. Le serie armoniche generalizzate

6.3. Definizioni e risultati teorici sulle serie

6.4. Confronto tra serie ed integrali impropri

8. Testi consigliati e bibliografia:

M. Bramanti, C.D. Pagani, S. Salsa "Analisi Matematica 1" © Zanichelli Editore

W. Dambrosio "Analisi Matematica" © Zanichelli Editore

Insegnamento **MFN0586 - Architettura degli Elaboratori**

Insegnamento (inglese):	**Computer architecture**
CFU:	**9**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**A - di base**
Docenti:	**Marco ALDINUCCI (Titolare)** **Rossano GAETA (Titolare)** **Michele GARETTO (Titolare)** **Maurizio LUCENTEFORTE (Titolare)** **Claudio SCHIFANELLA (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Conoscenza di base del linguaggio Java, nozioni di programmazione, Algebra di Boole.

Eventuali corsi propedeutici

È richiesta la conoscenza delle competenze fornite dai corsi di:

- Programmazione I e Laboratorio (I semestre)
- Matematica Discreta e Logica (I semestre)

2. Obiettivi formativi:

L’insegnamento si propone di fornire allo studente:

- la conoscenza delle tecniche di codifica dell’informazione all’interno dei calcolatori;
- la conoscenza dell’organizzazione hardware degli elaboratori, attraverso la nozione di gerarchie di macchine virtuali;
- la comprensione delle funzioni svolte dall’hardware e utilizzate dai sistemi operativi;
- le nozioni di base per la comprensione del processo di traduzione da linguaggi ad alto livello a linguaggio macchina;
- le conoscenze delle tecniche di microprogrammazione e di programmazione in un linguaggio assemblativo.

3. Risultati dell'apprendimento attesi:

Al termine dell’insegnamento lo studente dovrà dimostrare di essere in grado di:

- codificare e decodificare numeri con e senza segno, interi, frazionari, in virgola mobile;
- conoscere e analizzare circuiti logici elementari;
- saper analizzare i componenti digitali che sono alla base dell’architettura dei moderni calcolatori;
- conoscere e saper utilizzare le tecniche della microprogrammazione per l’esecuzione di istruzioni a livello macchina;
- conoscere le relazioni tra linguaggi ad alto livello (Java) ed equivalente linguaggio a livello macchina;
- saper sviluppare piccoli programmi in un linguaggio assemblativo e conoscere le relative tecniche di traduzione in linguaggio macchina e del collegamento (linking/loading).

4. Modalità di verifica dell'apprendimento:

L’esame consiste di una prova scritta con voto finale espresso in trentesimi (con possibilità di lode). La prova scritta è articolata in due parti: una relativa agli argomenti di teoria ed una relativa

agli argomenti di laboratorio. Ognuna delle due parti prevede un punteggio espresso in trentesimi ed il voto finale è definito come la media (ponderata rispetto ai CFU della teoria e del laboratorio) della valutazione delle due componenti. Le domande di teoria possono essere a risposta aperta oppure a risposta chiusa. Le domande a risposta aperta possono richiedere l'enunciazione e la descrizione di proprietà o di tecniche viste a lezione. Possono altresì richiedere l'applicazione della teoria a particolari esempi. Con queste domande si intende valutare la comprensione dei principi visti a lezione, la capacità di enunciarli con l'uso appropriato di un linguaggio scientifico e la capacità di applicarli a casi pratici. Le domande di laboratorio vertono sulla microprogrammazione e sulla programmazione in livello assemblativo.

5. Modalità d'insegnamento:

L'insegnamento è diviso in una parte di teoria e una di laboratorio. Per la parte di teoria sono previste 48 ore lezioni frontali, integrate da esempi e da esercitazioni. Per la parte di laboratorio sono previste 30 ore di attività in laboratorio in cui, con l'ausilio di appositi simulatori e applicativi, si svolgono esercizi e approfondimenti sugli argomenti del corso. La frequenza è facoltativa, consigliata, e la prova finale sarà uguale per frequentanti e non.

6. Attività di supporto:

Il materiale didattico di supporto sia per il corso A che per il corso B (lucidi, link, esempi di testo di esami ed altro) è disponibile presso il supporto on-line ai corsi I-learn

http://i-learn.educ.di.unito.it/.

Testi e soluzioni di alcuni esercizi svolti in aula possono essere trovati nelle pagine del corso di Architettura degli Elaboratori per gli anni accademici precedenti.

7. Programma:

- Introduzione all'organizzazione strutturata dei calcolatori: macchina di Von Neumann, macchine virtuali;
- Codifica dell'informazione: numeri binari, conversione tra basi, numeri negativi, operazioni tra numeri binari, numeri floating point e standard IEEE 754;
- Livello logico digitale: porte logiche e algebra di Boole, circuiti logici, latch, flip-flop, registri, chip di memoria, RAM e ROM;
- Memoria e organizzazione della memoria: organizzazione della memoria, memoria cache, memorie permanenti;
- Il bus;
- Il livello della microarchitettura: organizzazione della CPU ed esecuzione delle istruzioni, esempio di microarchitettura, l'ISA IJVM, il microprogramma per l'architettura Mic-1, ottimizzazione dell'architettura Mic-1 (Mic-2) ed un'architettura con pipeline (Mic-3);
- Il livello ISA: sommario al livello ISA, caratteristiche, modelli di memoria, registri e istruzioni, modalità di indirizzamento, formati delle istruzioni, tipi di dati ed istruzioni nel livello ISA, controllo del flusso, I/O, interrupt, trap;
- Assembler, linker, loader;

In laboratorio saranno svolte esercitazioni sulle codifiche, sui circuiti logici, sulla microprogrammazione in Mic-1, sulla scrittura di programmi in linguaggio assemblativo IJVM, con l'ausilio di appositi simulatori e applicativi.

8. Testi consigliati e bibliografia:

Andrew S. Tanenbaum, Todd Austin Architettura dei calcolatori: un approccio strutturale. 6a edizione Pearson Education Italia, 2013. ISBN 9788871929620

oppure

Andrew S. Tanenbaum. Architettura dei calcolatori: un approccio strutturale. 5a Edizione. Pearson - Addison Wesley, 2006. ISBN 8871922719.

Insegnamento **MFN0602 - Basi di Dati**

Insegnamento (inglese):	
CFU:	**9**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Luca ANSELMA (Titolare)** **Ruggero Gaetano PENSA (Titolare)** **Gian Luca POZZATO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Lo studente deve avere familiarità con i concetti fondamentali della teoria degli insiemi e della logica. Deve aver acquisito capacità di progettare algoritmi fondamentali e programmare in linguaggi ad alto livello.

Eventuali corsi propedeutici

Insegnamenti di Logica, Matematica Discreta e Programmazione 1 e 2

2. Obiettivi formativi:

L'insegnamento è un'introduzione alle basi di dati e ai sistemi di gestione delle medesime (SGBD). Si propone perciò di fornire allo studente le prime competenze teoriche e pratiche sul modello relazionale, sulla progettazione di una base di dati e sull'architettura dei SGBD introducendo le componenti fondamentali. In aula saranno introdotti i concetti teorici di base. Gli studenti acquisiranno conoscenze (teoriche e pratiche) su: progetto di una base dati, modello Entità Associazioni (EA) (o Entity Relationship), modello relazionale, algebra relazionale e SQL, dipendenze funzionali e normalizzazione, elementi di architettura dei SGBD relazionali (concorrenza, ripristino e buffer management, dizionario dei dati, memorizzazione efficiente dei dati). In laboratorio gli studenti lavoreranno al progetto di una base dati e ad esercitazioni su casi di studio in SQL.

3. Risultati dell'apprendimento attesi:

Gli studenti devono aver assimilato i concetti:

- i fondamenti matematici del modello relazionale,
- le interrogazioni in algebra ed in calcolo relazionale e in SQL,
- i fondamenti della teoria della normalizzazione,
- le strutture dati per la memorizzazione efficiente dell'informazione.

Inoltre lo studente deve:

- saper analizzare i requisiti per la progettazione di una base di dati per rappresentarli in forma di diagrammi Entity Relationship o Entità Associazioni (EA) essere in grado di trasformare schemi EA in schemi relazionali,
- essere in grado di trasformare interrogazioni in forma testuale in interrogazioni SQL, essere in grado di eseguire semplici ristrutturazioni di schemi EA sulla base di dati quantitativi (ad esempio: numero stimato di record e criticità delle interrogazioni,
- aver compreso i concetti e l'uso di transazioni.

4. Modalità di verifica dell'apprendimento:

E' obbligatoria l'iscrizione all'ambiente di e-learning Moodle dove gli studenti troveranno forum di discussione, materiale didattico, compiti assegnati e quiz sugli argomenti delle lezioni preparati dai docenti.

L'esame consiste in una prova scritta ed in una discussione orale del progetto di laboratorio.

- Prova scritta: la prova prevede da 6 a 8 domande in tutto, al fine di verificare l'apprendimento dei concetti di base. Di queste, una o due domande riguardano argomenti di laboratorio (SQL), le restanti domande riguardano argomenti ed esercizi della parte di teoria. La valutazione è in trentesimi.

Il superamento dello scritto permette di accedere a una delle prove orali previste per l'appello in cui lo scritto è stato superato. Nel caso in cui questa seconda prova non venga superata, lo scritto dovrà essere ripetuto. Lo studente ha diritto al più a tre correzioni dello scritto in un anno accademico.

- Prova di laboratorio: viene assegnato un progetto (su argomenti di progettazione) che può essere svolto individualmente o in gruppi formati da 2 o 3 studenti. La discussione della prova di laboratorio è individuale e verte sull'intero progetto. La valutazione è in trentesimi. La parte di laboratorio riguardante l'SQL verrà invece valutata contestualmente allo scritto.

Il voto finale sarà la media pesata dei voti ottenuti nelle tre prove (teoria, SQL e orale di progettazione), valutate in trentesimi, essendo comunque necessario il raggiungimento della sufficienza nelle due prove di laboratorio (Progettazione ed SQL) e della votazione di 15/30 nella prova scritta.

5. Modalità d'insegnamento:

Lezioni frontali: 48 ore. Attività di laboratorio: 30 ore. La frequenza costante è caldamente consigliata.

6. Attività di supporto:

E' obbligatoria l'iscrizione all'ambiente di e-learning Moodle dove gli studenti troveranno forum di dicussione, materiale didattico e compiti assegnati. E' altresì obbligatoria la consultazione di tali materiali.

7. Programma:

Funzionalità e componenti dei sistemi di gestione di basi di dati. Fondamenti teorici delle basi di dati relazionali:

- il modello relazionale delle basi di dati (definizioni, proprietà principali, vincoli di integrità),
- algebra relazionale,
- introduzione al calcolo relazionale,
- introduzione a SQL (DDL e DML), Dizionario dei dati in un DBMS
- dipendenze funzionali e teoria della normalizzazione,
- memorizzazione efficiente dei dati (B+ alberi),
- cenni alle tecniche di ottimizzazione,
- Introduzione alle transazioni: problemi di concorrenza e di affidabilità, livelli di isolamento.

Progettazione e programmazione delle basi dati:

- Specifica d'interrogazioni e realizzazione in SQL (con esercitazioni in laboratorio su ORACLE/Postgresql).
- Dizionario dei dati (con esercitazioni in laboratorio su ORACLE/Postgresql).
- Definizione e ruolo di "database administrator".
- Cenni ai meccanismi d'autorizzazione offerti da SQL (possibilmente con esercitazioni in laboratorio su ORACLE/Postgresql).
- Introduzione alle metodologie di progettazione del software e loro relazione con la progettazione della basi di dati, argomenti non trattati e relazione con altri insegnamenti (es. Ingegneria del Software).

- Progettazione concettuale/logica, usando il modello ER (Entity Relationship) con eventuali esercizi di reverse modelling.
- Considerazioni sui parametri quantitativi dello schema logico.
- Considerazioni su meccanismi d'indicizzazione.

8. Testi consigliati e bibliografia:

Testo di riferimento: Atzeni, Ceri, Fraternali, Paraboschi, Torlone, "Basi di dati", McGraw-Hill, Quinta edizione, 2018.

Materiali aggiuntivi sono forniti dai docenti.

Insegnamento **INF0090 - Calcolabilità e Complessità**

Insegnamento (inglese):	**Computability and Complexity**
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**0**
Tipologia di Attività Formativa:	**B - caratterizzante** **D - libera**
Docenti:	**Stefano BERARDI (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Sono richieste buone conoscenze di logica, programmazione e di algoritmi; inoltre si assume che lo studente possegga le nozioni di linguaggio formale, grammatica e di automa.

Eventuali corsi propedeutici

Matematica Discreta e Logica, Programmazione 1 e 2, Algoritmi e Strutture Dati, Linguaggi Formali e Traduttori.

2. Obiettivi formativi:

Che cos'e' un algoritmo? Quali problemi si possono risolvere con un algoritmo? E in quali casi un algoritmo richiede risorse inaccessibili nella pratica? Il corso affronta questi problemi, trattando anzitutto la teoria della computabilita' sia dal punto di vista matematico - macchine di Turing, funzioni ricorsive - che da prospettive legate ai linguaggi di programmazione, come quella dei programmi while. Si discutono poi i vari possibili criteri di misura delle risorse disponibili (tempo, memoria, cpu) e le classi di complessita'. Finiamo presentando il problema P = NP .

3. Risultati dell'apprendimento attesi:

Conoscenza delle definizione del concetto di calcolo, una maggiore consapevolezza delle limitazioni intrinseche all'uso delle macchine ed un'idea delle strategie per sopperire a tali limitazioni.

4. Modalità di verifica dell'apprendimento:

Esame orale.

5. Modalità d'insegnamento:

Le lezioni si svolgono in aula, ed utilizzano la piattaforma Moodle sia per inviare messaggi agli studenti che per le verifiche dell'apprendimento durante il corso.

6. Attività di supporto:

Vedi il sito moodle del corso

7. Programma:

Teoria della computabilita'

- Le Macchine di Turing
- Problemi non risolubili
- Funzioni ricorsive
- Calcolabilita' e Linguaggi di Programmazione

Teoria della complessita'

- Misure e classi di Complessita'
- Classi di Complessita' Temporale
- Classi di Complessita' Spaziale
- Le classi P ed NP
- Problemi NP completi

8. Testi consigliati e bibliografia:

L'**unico libro di testo del corso** e' il seguente: ***M. Sipser:"Introduction to the theory of Computation", Course Technology Ptr., 3^edition, 2012.*** Le sezioni effettivamente svolte verranno indicate alla fine dell'anno.

(Per chi desiderasse rivedere in italiano i concetti del testo inglese, segnaliamo il testo: C. Toffalori et alii, Teoria della calcolabilita' e della complessita', McGraw-Hill 2005. Ribadiamo che, tuttavia, questo testo non viene utilizzato nel corso.)

Insegnamento **MFN0612 - Calcolabilità e Complessità A**

Insegnamento (inglese):	**Computability and Complexity - A**
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**B - caratterizzante** **D - libera**
Docenti:	**Stefano BERARDI (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Sono richieste buone conoscenze di logica, programmazione e di algoritmi; inoltre si assume che lo studente possegga le nozioni di linguaggio formale, grammatica e di automa.

Eventuali corsi propedeutici

Matematica Discreta e Logica, Programmazione 1 e 2, Algoritmi e Strutture Dati, Linguaggi Formali e Traduttori.

2. Obiettivi formativi:

Che cos'e' un algoritmo? Quali problemi si possono risolvere con un algoritmo? E in quali casi un algoritmo richiede risorse inaccessibili nella pratica? Il corso affronta questi problemi, trattando anzitutto la teoria della computabilita' sia dal punto di vista matematico - macchine di Turing, funzioni ricorsive - che da prospettive legate ai linguaggi di programmazione, come quella dei programmi while. Si discutono poi i vari possibili criteri di misura delle risorse disponibili (tempo, memoria, cpu) e le classi di complessita'. Finiamo presentando il problema P = NP .

3. Risultati dell'apprendimento attesi:

Conoscenza delle definizione del concetto di calcolo, una maggiore consapevolezza delle limitazioni intrinseche all'uso delle macchine ed un'idea delle strategie per sopperire a tali limitazioni.

4. Modalità di verifica dell'apprendimento:

Esame orale.

5. Modalità d'insegnamento:

Le lezioni si svolgono in aula, ed utilizzano la piattaforma Moodle sia per inviare messaggi agli studenti che per le verifiche dell'apprendimento durante il corso.

6. Attività di supporto:

Vedi il sito moodle del corso

7. Programma:

Teoria della computabilita'

- Le Macchine di Turing
- Problemi non risolubili
- Funzioni ricorsive
- Calcolabilita' e Linguaggi di Programmazione

Teoria della complessita'

- Misure e classi di Complessita'
- Classi di Complessita' Temporale
- Classi di Complessita' Spaziale
- Le classi P ed NP
- Problemi NP completi

8. Testi consigliati e bibliografia:

L'**unico libro di testo del corso** e' il seguente: ***M. Sipser:"Introduction to the theory of Computation", Course Technology Ptr., 3^edition, 2012.***

Svolgeremo circa **1/3 del libro**, scegliendo sezioni dalla **parte 2 (calcolabilita')** e dalla **parte 3 (complessita')**. Le sezioni effettivamente svolte ***verranno indicate alla fine di ogni lezione e del corso***.

(Per chi desiderasse rivedere in italiano i concetti del testo inglese, segnaliamo il testo: C. Toffalori et alii, Teoria della calcolabilita' e della complessita', McGraw-Hill 2005. Ribadiamo che, tuttavia, questo testo non viene utilizzato nel corso.)

Insegnamento	**MFN0588 - Calcolo Matriciale e Ricerca Operativa**

Insegnamento (inglese):	**Matrix Calculus and Operational Research**
CFU:	**6**
Settore:	**MAT/09 - RICERCA OPERATIVA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**A - di base**
Docenti:	**Roberto ARINGHIERI (Titolare)** **Andrea Cesare GROSSO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Nozioni di base di algebra e insiemistica comuni nei programmi di matematica della scuola superiore.

Eventuali corsi propedeutici

Nessuno.

2. Obiettivi formativi:

Il corso si propone di fornire agli studenti nozioni generali di calcolo matriciale, algebra e geometria, e nozioni più specifiche di ricerca operativa.

Il calcolo matriciale è uno strumento fondamentale per il calcolo scientifico. La ricerca operativa studia modelli e metodi, basati sulle tecniche introdotte, per l'utilizzo ottimale di risorse scarse (in ambiti produttivi, finanziari, ecc.).

3. Risultati dell'apprendimento attesi:

Lo studente deve apprendere nozioni di geometria di base e le tecniche essenziali per manipolare vettori e matrici in spazi a n dimensioni.

Lo studente deve acquisire la capacità costruire modelli di programmazione lineare sia a variabili continue che a variabili intere partendo dall'enunciato di un problema reale, e la conoscenza dei meccanismi di base e la teoria relativa agli algoritmi che operano su tali modelli.

4. Modalità di verifica dell'apprendimento:

L’esame è costituito da una prova scritta di durata di almeno 2 ore seguita da una prova orale facoltativa.

Regole da seguire durante la partecipazione ad una prova scritta:

- E' vietata la comunicazione, sotto ogni forma, sia tra persone in aula che fuori dall’aula. La violazione di questa regola comporta l’annullamento della prova.
- E’ permesso portare come materiale il formulario del corso.
- E' vietato portare ed utilizzare gli appunti.
- E’ permesso portare la calcolatrice.
- Autovalutazione in 3 passi:
    1. Le soluzioni dell'esame saranno rese disponibili on-line subito dopo la prova scritta.
    2. Lo studente che ha consegnato l'esame ha 24 ore di tempo per decidere di ritirare la propria prova.
    3. Gli studenti che non ritirano lo scritto consegnato, in caso di prova non superata

(ovvero voto<15), consumano 1 delle 3 consegne a disposizione sui 5 appelli disponibili.

Validità dei risultati ottenuti durante le prove che formano l’esame:

- Il voto ottenuto durante una prova rimane valido durante tutto l’Anno Accademico in cui la prova è stata sostenuta.
- La ripetizione di una prova, ovvero presenza effettiva all'appello anche in caso di ritiro, comporta l’annullamento dell’esito della prova precedente.

Regole per il calcolo del voto di un esame:

- La prova scritta viene valutata da 0 a 33 e si considera superata con voto uguali o superiori a 15
- La prova orale viene valutata da -4 a 4.
- Il voto dell’esame si ottiene come somma dei voti ottenuti nelle prove che lo compongono.

5. Modalità d'insegnamento:

Lucidi proiettati in aula e tradizionali spiegazioni alla lavagna.

6. Attività di supporto:

Verranno forniti appunti ed eserciziari, disponibili sulla pagina I-learn del corso.

7. Programma:

1. Vettori e matrici. Operazioni fondamentali. Insiemi convessi, poliedri.
2. Soluzione di un sistema di equazioni lineari.
3. Combinazioni lineari, indipendenza lineare.
4. Programmazione lineare.
    - Modellazione.
    - Struttura della regione ammissibile. Metodo grafico di soluzione.
    - Soluzioni di base. Algoritmo del simplesso.
5. Algoritmo di Branch&Bound per problemi di programmazione lineare intera.

8. Testi consigliati e bibliografia:

R.J. Vanderbei, "Linear Programming", Kluwer Academic Publishers (solo per eventuali approfondimenti - reperibile in biblioteca)

Insegnamento	**MFN0604 - Economia e Gestione dell'Impresa e Diritto**

Insegnamento (inglese):	
CFU:	**9**
Settore:	**IUS/02 - DIRITTO PRIVATO COMPARATO** **SECS-P/08 - ECONOMIA E GESTIONE DELLE IMPRESE**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**C - affine e integrativa**
Docenti:	**Fabio MONTALCINI (Professore a Contratto)** **Marco PIRONTI (Titolare)** **Camillo SACCHETTO (Professore a Contratto)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Il corso non presuppone conoscenze iniziali specifiche

Eventuali corsi propedeutici

NESSUNO

2. Obiettivi formativi:

Modulo di Economia

l'obiettivo del corso è analizzare le caratteristiche basi di una azienda: dal modello di business teorico alla creazione dei processi produttivi, alla comunicazione e vendita dei prodotti.

Verranno studiate le relazioni delle aziende all'interno della catena del valore (Clieti /fornitori) e i loro settori di riferimento. I framework teorici saranno poi applicati per l'analisi di aziende, settori e mercati innovativi al fine di valutare come vengono colte le opportunità che l'innovazione dà alle aziende e ipotizzare sviluppi futuri di modelli. durante il corso gli studenti si confronteranno con casi aziendali reali e con imprenditori. gli studeneti del corso potranno inoltre partecipare al progetto silicon valley study tour e passare una settimana in Silicon Valley dove conoscere le aziende più innovative dell'IT.

Gli studenti hanno inoltre l'opportunità di partecipare: www.ggi-academy.it/

Modulo di Diritto

Agli studenti saranno forniti le nozioni, i metodi, i modelli cognitivi e gli strumenti tecnici necessari alla comprensione dei concetti fondamentali del Diritto dell'Informatica, settore della scienza forense che realizza maggiormente il dialogo teorico e la sintesi operativa tra il diritto (in particolar modo nei suoi profili pratici e processuali) e l'informatica. Durante il Corso si analizzeranno i principali temi della materia anche alla luce dei piu' significativi casi nazionali e internazionali.

3. Risultati dell'apprendimento attesi:

Modulo di Economia

CREAZIONE E ANALISI DEI MODELLI DI BUSINESS CREAZIONE DELLE STRATEGIE AZIENDALI ANALISI DEI COMPETITOR, DEL SETTORE, DEL MERCATO.

SOFT SKILL: LAVORO IN TEAM CAPACITA' DI PRESENTARE PROBLEM SOLVING

Modulo di Diritto

Lo studente apprenderà il know-how normativo, concettuale ed operativo (sostanziale e processuale) per l'analisi e la verifica delle principali fattispecie giuridiche inerenti le nuove tecnologie dell'informazione attraverso lo studio delle sentenze, dei casi pratici e delle regole concrete fondamentali relative ai principali procedimenti – nazionali ed internazionali – nei quali sono coinvolti l'Informatica ed il Diritto.

4. Modalità di verifica dell'apprendimento:

Modulo di Economia

70% esame scritto sul materiale delle lezioni 30% lavoro di gruppo i non frequentanti: 70% sul materiale del sito+ libri 30% lavoro individuale.

Modulo di Diritto

L'apprendimento degli argomenti trattati nel corso sarà valutato mediante una verifica in cui saranno accertate le conoscenze degli aspetti informatico giuridici affrontati.

5. Modalità d'insegnamento:

frontali

6. Attività di supporto:

7. Programma:

Modulo di Economia

business model struttura dell'organizzazione struttura dei processi struttura delle funzioni strategie analisi di settore analisi di mercato

Modulo di Diritto

Responsabilità Internet Service Provider; Computer Crimes e Digital Forensics; Documento informatico, Firme elettroniche e Posta Elettronica Certificata (PEC); Proprietà intellettuale in ambiente web; Privacy: Principi generali e suoi ambiti peculiari; E-commerce e web contracts.

8. Testi consigliati e bibliografia:

Modulo di Economia

Creare modelli di business. Un manuale pratico ed efficace per ispirare chi deve creare o innovare un modello di business Autore Osterwalder Alexander; Pigneur Yves

Modulo di Diritto

I materiali didattici saranno suggeriti e forniti dai docenti durante il corso.

Insegnamento

MFN0617 - Economia e Gestione dell'Innovazione

Insegnamento (inglese):	
CFU:	**6**
Settore:	**SECS-P/08 - ECONOMIA E GESTIONE DELLE IMPRESE**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**D - libera**
Docenti:	**Marco PIRONTI (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Conoscenza dei concetti base di Economia e Gestione delle Imprese: analisi di settore analisi di mercato/consumatore modello di business analisi delle organizzazioni strategie delle aziende

Eventuali corsi propedeutici

Corso di Economia e Gestione delle Imprese

2. Obiettivi formativi:

Questo corso ha l'obiettivo di fornire gli skill, tools e approcci per la creazione di un' innovazione sostenibile e di successo nell'era digitale. il corso è multidisciplinare e costruito sulle ultime ricerche legate ai framework di gestione dell'innovazione, sulle metodologie lean per la creazione dei prototipi, sull'additive manufacturing e sull'approccio del design thinking. Insieme, esploreremo come identificare le opportunità dell'economia digitale e creare un'innovazione che va ben oltre il semplice ricopiare o reinventare un prodotto o servizio risolvendo problemi reali della nostra società quali creazione di lavoro, diminuzione di side effect eliminazione di colli di bottiglia all'interno di prodotti, modelli e settori. identificheremo inoltre framework informatici per facilitare l'identificazione,lo sviluppo e la creazione di innovazione in modo economico e sostenibile. l'obiettivo del corso non sarà solo acquistare gli strumenti teorici per creare un innovazione ma anche riuscire ad applicarli per arrivare a creare un'innovazione di prodotto o servizio all'interno di un mercato di riferimento Gli studenti del corso parteciperanno alla European innovation Accademy

3. Risultati dell'apprendimento attesi:

capacità di gestire un processo di creazione dell'innovazione capacità di utilizzare diversi approcci quali: 1)job to be done per il riconoscimento dell'innovazione e dei clienti 2)lean methodology: per la creazione del MINIMUM VIABLE PRODUCT e i testing dei prototipy 3)design thinking: per la comunicazione del prodotto e strumenti

4. Modalità di verifica dell'apprendimento:

le modalità d'esame sono differenti a seconda che lo studente sia frequentante o non frequentate. 1) modalità di esame per i frequentanti 70% valutazione attraverso una prova scritta 30% progetto di gruppo e partecipazione attiva in aula 2)non frequentanti: 70% valutazione attraverso una prova scritta 30%progetto individuale

5. Modalità d'insegnamento:

Le elzioni si basano su lezioni teoriche e laboratori pratici di: tecnologia per la costruzione di prototipi lato software/hardware software di statistica per la definizione dei segmenti di mercato (cluster analysis)

6. Attività di supporto:

le slide del corso verranno inserite sul sito del corso

7. Programma:

---il consumatore ---l'individuazione dell'opportunità ---la value proposition ---creazione del prototipo --Test Lean ---sperimentazione ---design/comunicazione e visualizzazione

8. Testi consigliati e bibliografia:

"what customers want" Tony Ulwich

Insegnamento **MFN0600 - Elementi di Probabilità e Statistica**

Insegnamento (inglese):	**Foundations of Probability and Statistics**
CFU:	**6**
Settore:	**MAT/06 - PROBABILITA\' E STATISTICA MATEMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**C - affine e integrativa**
Docenti:	**Roberta SIROVICH (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Calcolo di base.

2. Obiettivi formativi:

L’insegnamento si inserisce nell’area tematica della matematica, in particolare della probabilità e statistica. Si propone il duplice obiettivo di fornire allo studente sia la conoscenza e la capacità di comprensione dei fenomeni di natura aleatoria sia gli strumenti metodologici e analitici correlati, che siano di supporto per i corsi successivi ma anche di valore intrinseco. L’introduzione ai concetti di rischio e di probabilità fornisce gli strumenti analitici e modellistici per la trattazione di eventi casuali. L’introduzione alla statistica fornisce gli strumenti metodologici per trattare con le quantità aleatorie rilevabili.

3. Risultati dell'apprendimento attesi:

Lo studente avrà familiarità con il calcolo della probabilità di eventi, il concetto di variabile aleatoria e con le principali distribuzioni di probabilità. Avrà acquisito i concetti base della statistica descrittiva e inferenziale. Sarà inoltre in grado di svolgere le principali analisi dati considerate standard. Lo studente a tale scopo apprenderà l’utilizzo del software R.

4. Modalità di verifica dell'apprendimento:

La verifica della preparazione dello studente avviene con un esame scritto. L’esame è svolto in laboratorio sotto forma di un quiz Moodle.

5. Modalità d'insegnamento:

Lezioni ed esercitazioni frontali in aula.

6. Attività di supporto:

7. Programma:

Il programma di lavoro è articolato in due parti, la probabilità e la statistica.
Probabilità. Spazio campionario e probabilità, insiemi, modelli probabilistici, probabilità condizionata, teorema delle probabilità totali, formula di Bayes, indipendenza. Variabili aleatorie discrete, funzione di densità discreta, funzioni di variabili aleatorie, attesa, varianza. Densità discreta congiunta di variabili multidimensionali, condizionamento e indipendenza. Variabili aleatorie continue, funzione di densità, funzione di distribuzione cumulata. Densità di probabilità variabili aleatorie multidimensionali. Covarianza e correlazione. Legge dei grandi numeri e teorema del limite centrale.
Statistica. Introduzione a R. Dati univariati, tipi di dati, indici riassuntivi di posizione, di variabilità e di forma. Rappresentazioni grafiche, istogrammi e box plot. Dati invariati, scatterplot. Confronti qualitativi, dati appaiati. Dati invariati categoriali e tabelle. Intervalli di confidenza (proporzioni,

medie, varianze, differenze e mediane). Test di ipotesi (proporzioni, medie, varianze, mediane). Test per due campioni. Test di bontà del fit. Regressione lineare. Analisi della varianza.

8. Testi consigliati e bibliografia:

DP Bertsekas and JN Tsitsiklis. *Introduction to probability*. 2nd Edition. Athena Scientific, 2008.
J Verzani. *Using R for Introductory Statistics*. 2nd Edition. CRC Press, 2014.

Insegnamento	**MFN0598 - Fisica**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**FIS/01 - FISICA SPERIMENTALE**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**C - affine e integrativa**
Docenti:	**Martino GAGLIARDI (Titolare)** **Ernesto MIGLIORE (Titolare)** **Igor PESANDO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Competenze di base di trigonometria, di calcolo vettoriale e di analisi matematica.

Eventuali corsi propedeutici

Corsi di Matematica del I anno.
Nella prima settimana di lezione si svolgera` un ciclo di lezioni (8h) di richiamo sulla *Meccanica del punto materiale* (argomenti trattati: richiami di cinematica; leggi di Newton; lavoro ed energia; forze conservative).

2. Obiettivi formativi:

L'insegnamento si propone di:

1. introdurre alla conoscenza della basi indispensabili di meccanica, delle principali proprietà del campo elettrico e del campo magnetico, con cenni al comportamento della materia soggetta a tali campi;
2. introdurre alla conoscenza del comportamento degli elementi di un circuito in corrente continua ed in corrente alternata;
3. introdurre alla conoscenza dei principi fisici alla base del funzionamento delle porte logiche.

3. Risultati dell'apprendimento attesi:

Lo studente deve avere acquisito la capacità di esprimere i concetti con un formalismo matematico sufficientemente rigoroso e di collegare fra di loro in modo armonico i principi fondamentali della fisica. Deve inoltre essere in grado di fornire una risposta quantitativa ai problemi assegnati.

4. Modalità di verifica dell'apprendimento:

L'esame di Fisica è costituito dalla sola prova scritta. Il formato dell'esame (2h a disposizione) è il seguente:

1. 1 esercizio sull'algebra dei vettori
2. 1 esercizio su campo elettrico oppure su campo magnetico + 1 esercizio su circuiti
3. 1 domanda aperta da svolgere (max 1 facciata) tra 2 estratte, al momento dello scritto, da questa lista

Per superare l'esame occorre soddisfare le seguenti condizioni (operazione logica AND fra le condizioni):

- avere fatto tutto giusto l'esercizio sui vettori
- avere svolto >50% complessivo dei due esercizi su campi e circuiti
- avere risposto in modo esaustivo alla domanda aperta

Il primo esercizio e` di sbarramento e **non** conta per il voto finale. La restante parte di esercizi e la parte di teoria contribuiscono in **egual** misura alla determinazione del voto finale.
Il >50% complessivo dei due esercizi puo' esser ottenuto per esempio con 10%+45% o 30%+30% dove il primo valore percentuale si riferisce all'esercizio sull'elettrmagnetismo ed il secondo all'esercizio sui circuiti.

All'esame e` possibile utilizzare la calcolatrice.
Non e` consentito l'uso di libri o appunti ma solo di un formulario distribuito dai docenti in sede di esame. Gli scritti assegnati negli ultimi anni e le relative soluzioni sono reperibili al seguente link

5. Modalità d'insegnamento:

L'insegnamento si svolge attraverso lezioni frontali ed esercitazioni svolte dagli stessi docenti. Nella prima settimana di lezione e` previsto un ciclo di lezioni (8h) di richiamo sulla *Meccanica del punto materiale*.

6. Attività di supporto:

Supporto tramite web
L'insegnamento e` presente sulla piattaforma di e-learning. Il forum dell'insegnamento e` usato abitualmente per le comunicazioni con gli studenti e per la distribuzione del materiale didattico. Il materiale relativo alle esercitazioni e` reperibile al link: http://personalpages.to.infn.it/~gagliard/

Tutoraggio in aula
Vengono fornite ore di tutoraggio (2 ore/settimana) in cui si ha la possibilità di svolgere gli esercizi sotto la guida di un tutore. Gli esercizi proposti sono scelti in stretto rapporto del docente dell'insegnamento con il tutore.

TUTORATO a.a. 2018-2019
Tutorato, II semestre (consultare l'orario).

7. Programma:

Il campo elettrostatico. Teorema di Gauss. Conservatività del campo elettrostatico. Superfici equipotenziali. Conduttori e dielettrici. Capacità elettrica di un conduttore. Condensatori. Densità di energia del campo elettrico. Correnti elettriche. Leggi di Ohm e di Kirchhoff. Circuiti RC.

Fisica dei dispositivi elettronici. Conduttori, isolanti e semiconduttori. Giunzione pn e diodo. Dispositivi MOS e porte logiche (NOT, NOR, NAND).

Il campo magnetico indipendente dal tempo. Magneti. Moto di una carica in campo magnetico; esempi ed applicazioni. Filo percorso da corrente in campo magnetico. Campo magnetico generato da un filo percorso da corrente. Teorema di Ampere. Teorema di Gauss per il campo magnetico.

Campi elettrici e magnetici variabili nel tempo. Induzione elettromagnetica. Legge di Faraday-Henry. Correnti alternate. Legge di Ampere-Maxwell. Autoinduzione. Induttanza del solenoide ideale. Densità di energia del campo magnetico. Circuiti RL. Elementi circuitali in corrente alternata.

Il programma dettagliato relativo all'insegnamento sara` pubblicato dai docenti al termine delle lezioni.

E` possibile consultare anche il registro delle lezioni dell'A.A. 2017-2018:

- programma dettagliato corso A (A.A. 2017/2018) pdf
- programma dettagliato corso B (A.A. 2012/2013) pdf

8. Testi consigliati e bibliografia:

Testi consigliati (tra cui scegliere):

1. WOLFSON, Fisica, vol.2 - Elettromagnetismo, ottica e fisica moderna, Pearson
2. RESNICK/HALLIDAY/KRANE, Fisica, vol. 2, Casa Editrice Ambrosiana
3. SERWAY/JEWETT Fisica. Per scienze ed ingegneria, vol. 2, EdiSes

Insegnamento **MFN1353 - Interazione Uomo Macchina**

Insegnamento (inglese):	
CFU:	**6**
Settore:	
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**D - libera**
Docenti:	**Viviana PATTI (Titolare)** **Marino SEGNAN (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Si richiede una buona conoscenza delle basi di dati (fornita dal corso di Basi di Dati), dell'analisi e della progettazione object-oriented (fornita dai corsi di Programmazione II e di Algoritmi e strutture dati) e dei fondamenti della programmazione distribuita (fornita dal corso di Programmazione III). Lo studente deve inoltre avere la capacità di scrivere, compilare e verificare la correttezza di programmi in Java.

Eventuali corsi propedeutici

Basi di dati e sperimentazioni, Algoritmi e sperimentazioni, Programmazione II and III.

2. Obiettivi formativi:

Nella progettazione e sviluppo di un'applicazione software si devono tenere in conto due aspetti fondamentali: (i) l'interazione con l'utente che, indipendentemente dall'efficacia delle funzionalità offerte da un'applicazione, influenza sensibilmente il suo successo in termini di utilizzo. L'applicazione deve essere usabile ed accessibile per permettere un uso agevole del software e prevenirne l'abbandono da parte degli utenti. (ii) l'implementazione dell'applicativo, che richiede di guardare "dietro all'interfaccia utente" per andare a fondo su aspetti architetturali e tecnologici che possono influenzare non solo le prestazioni dell'applicazione, la sua scalabilità e robustezza, ma anche le tipologie di servizio che possono effettivamente essere offerte.

Partendo da queste considerazioni, il corso si pone un duplice obiettivo: da una parte, fornire la conoscenza di base necessaria per il disegno corretto di interazioni uomo-macchina, che sta alla base della progettazione di applicazioni di ogni genere (web e non, mobili, etc.). Dall'altra, concentrandosi sulle applicazioni mobili, che sono oggetto specifico del corso, fornire la conoscenza di base necessaria per la progettazione e lo sviluppo di applicazioni interattive, accessibili da terminali mobili (come per esempio gli smart phone), e caratterizzate da una logica applicativa mediamente complessa.

Più precisamente, per quanto riguarda l'interazione uomo-macchina, verranno acquisite competenze sia teoriche sia pratiche nel disegno corretto di interazioni, con specifico riferimento alla progettazione user-centered. Per quanto riguarda gli aspetti architetturali e tecnologici, il corso tratterà dal punto di vista sia teorico che pratico la programmazione lato client per device mobili su piattaforma Android e lo sviluppo di interfacce mobili. Per permettere agli studenti di sperimentare le nozioni apprese durante le ore di teoria in aula il corso include una sostanziale parte di laboratorio. I temi introdotti durante il laboratorio corredano e integrano le conoscenze derivanti dalla parte teorica (knowledge and understanding) e permettono agli studenti di familiarizzare con le metodologie e tecnologie introdotte, anche investigando soluzioni alternative (applying knowledge and understanding).

Inoltre durante le ore di laboratorio è previsto lo sviluppo di un'applicazione realistica con interfaccia mobile ed una introduzione al linguaggio Python. La preparazione e la discussione del progetto di laboratorio sono inoltre volte a stimolare le capacità di organizzare il lavoro in piccoli

gruppi (max 4 studenti), e poi di illustrare verbalmente le soluzioni adottate (communication skills).

3. Risultati dell'apprendimento attesi:

Lo studente dovrà avere assimilato: la progettazione user-centered, la prototipazione, la valutazione dell'usabilità e il disegno inclusivo per categorie speciali di utenti. Inoltre dovrà avere assimilato i concetti rilevanti per analizzare l'interazione: affordance, modelli concettuali, metafore, feedback, etc.

Lo studente acquisirà anche la conoscenza delle varie architetture di riferimento per lo sviluppo di applicazioni web e mobile, e dei loro pro e contro; inoltre lo studente acquisirà la conoscenza dei modelli più comunemente adottati per gestire dialoghi mediamente complessi e articolati tra utente e applicazione, e delle tecnologie attualmente utilizzate per l'implementazione delle applicazioni e delle loro interfacce (web e mobile). Lo studente dovrà essere in grado di sviluppare applicazioni lato client in ambiente Android e sviluppare un programma scritto in linguaggio Python.

4. Modalità di verifica dell'apprendimento:

L'esame è composto da un TEST SCRITTO e da una VERIFICA DI LABORATORIO. Le due prove possono essere sostenute in qualsiasi ordine (cioè, non è necessario aver dato lo scritto per fare la prova di laboratorio, né il vice versa).

TEST SCRITTO: Prova scritta che include esercizi e domande teoriche sul programma del corso. Viene valutata da un minimo di 0 ad un massimo di 15 e si considerano sufficienti i voti >=9. Durante la prova è proibito comunicare con altre persone, presenti in aula o fuori. Inoltre, non si può portare alcun tipo di materiale didattico (appunti, libri, dispense, etc.) ed è vietato usare computer, telefonini o simili. Come da regolamento di Ateneo, ogni studente può sostenere un numero massimo di tre prove scritte durante l'Anno Accademico (cioè, consegnare il proprio elaborato tre volte). Il voto ottenuto durante un test scritto decade se lo studente partecipa ad un altro test scritto e consegna il suo elaborato.

VERIFICA DI LABORATORIO: prevede la discussione del progetto di laboratorio svolto durante il corso. La discussione deve essere effettuata preferibilmente in unica soluzione, con tutti i membri del gruppo di laboratorio presenti. Il voto di laboratorio è un numero compreso tra 0 e 15, si considerano sufficienti i voti >=9. CALCOLO DEL VOTO FINALE DI ESAME: Sia X il voto del test scritto; sia Y il voto di laboratorio. Il voto Fin finale dell'esame si ottiene come segue: Fin = (X+Y)

Note: i voti acquisiti durante la prova di laboratorio, o durante il test scritto, rimangono validi fino al termine della terza sessione d'esame (quella che precede l'inizio del nuovo corso). Quando si superano entrambe le prove, è necessario registrare il voto finale entro i limiti imposti dal Regolamento di Ateneo.

5. Modalità d'insegnamento:

L'insegnamento è diviso in una parte di teoria e una di laboratorio. Per la parte di teoria sono previste 30 ore di lezione frontali che seguono il programma riportato più avanti, integrate da casi di studio e da esercitazioni volte ad illustrare l'applicazione pratica dei concetti appena studiati.

La parte di laboratorio consiste di 30 ore ed è focalizzata sulla programmazione client-side in ambiente Android utilizzando le API Java del sistema operativo. Verra' inoltre sviluppata un' applicazione scritta in linguaggio Python . Le lezioni si svolgono in maniera interattiva e sono corredate da vari esercizi miranti a fornire esempi pratici.

Le sperimentazioni che vengono effettuate durante le ore di laboratorio, strutturate come sequenze di esercizi specifici, sono fondamentali per aiutare gli studenti a comprendere e assimilare i contenuti teorici spiegati a lezione in quanto permettono di mettere in pratica i concetti e le metodologie illustrate su esempi concreti. Inoltre lo sviluppo del progetto di laboratorio permette di consolidare le conoscenze teoriche in un caso realistico di media complessità.

Si consiglia caldamente la frequenza costante alle lezioni di teoria e di laboratorio. E' inoltre

fondamentale che gli studenti si iscrivano al corso online su I-Learn, all’interno del quale i docenti mettono a disposizione materiale didattico di supporto.

6. Attività di supporto:

Vengono forniti on-line i lucidi delle lezioni, link a documentazione disponibile sul Web ed alcuni testi degli esami scritti degli anni precedenti.

Il materiale didattico di supporto (lucidi delle lezioni, link a documentazione, esempi di testo di esami ed altro) è disponibile presso il supporto on-line ai corsi I-learn .

Si noti che i lucidi non sostituiscono il libro di testo, ne' il materiale integrativo ad esso affiancato.

7. Programma:

- Parte Ia - Human-computer interaction (HCI)
    - Human-computer interaction (HCI): Definizioni e contesto, evoluzione di HCI, nuove direzioni.
    - Il fattore umano: percezione (gestalt e affordance), attenzione e memoria, modelli mentali, metafore, il modello di Shneiderman e il modello di Norman.
    - Disegno di interazioni: user-centered design, requisiti funzionali e di usabilità (raccolta, analisi, presentazione), prototipazione, linee guida (con gestione degli errori ed assistenza agli utenti), elementi di tipografia elettronica, di layout e gestione del colore.
    - Tecniche di valutazione: valutazione senza utenti (quantitativa e qualitativa), valutazione con utenti, problemi, presentazione dei risultati.
    - Disegno inclusivo: accessibilità, disegno per utenti di differenti gruppi di età (bambini, anziani), internazionalizzazione.
- Parte Ib - Programmazione di device mobili.
    - Introduzione alla programmazione per mobile.
    - La piattaforma Android e sua architettura.
    - Processi e applicazioni in Android.
    - Il linguaggio Python:differenze rispetto a Java
    - Progettazione di una interfaccia utente in maniera programmatica e dichiarativa.
    - Sviluppare con Python o Java? Confronto tra gli ambienti ed esempi
    - Esempio di sviluppo del lato client di una semplice app per Android.

8. Testi consigliati e bibliografia:

Libri di Testo:

- Polillo, R. - FACILE DA USARE, Edizioni Apogeo, 2010
- Programmazione Web lato Server, di V. Della Mea, L. Di Gaspero, I. Scagnetto, Apogeo, 2007
- Android Training. http://developer.android.com/training/index.html
- Python. http://pythonspot.com/beginner/

Altri testi (per consultazione):

- La caffettiera del masochista, di D. Norman, Apogeo

Insegnamento	**MFN0608 - Interazione Uomo Macchina e Tecnologie Web**

Insegnamento (inglese):	
CFU:	**12**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Liliana ARDISSONO (Titolare)** **Viviana PATTI (Titolare)** **Marino SEGNAN (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Si richiede una buona conoscenza delle basi di dati (fornita dall'insegnamento Basi di Dati), dei sistemi operativi (fornita dall'insegnamento Sistemi Operativi), dell'analisi e della progettazione object-oriented (fornita da Programmazione II e di Algoritmi e strutture dati) e dei fondamenti della programmazione distribuita (fornita dall'insegnamento Programmazione III). Gli studenti e le studentesse devono inoltre avere la capacità di scrivere, compilare e verificare la correttezza di programmi in Java.

Eventuali corsi propedeutici

Basi di Dati, Sistemi Operativi, Programmazione II, Algoritmi e strutture dati, Programmazione III

2. Obiettivi formativi:

Nella progettazione e sviluppo di un'applicazione software si devono tenere in conto due aspetti fondamentali: (i) l'interazione con l'utente, indipendentemente dall'efficacia delle funzionalità offerte da un'applicazione, influenza sensibilmente il suo successo in termini di utilizzo. L'applicazione deve essere usabile ed accessibile per permettere un uso agevole del software e prevenirne l'abbandono da parte degli/delle utenti. (ii) l'implementazione dell'applicativo richiede di guardare "dietro all'interfaccia utente" per andare a fondo su aspetti architetturali e tecnologici che possono influenzare non solo le prestazioni, la scalabilità e la robustezza, ma anche le tipologie di servizio che possono effettivamente essere offerte.

Partendo da queste premesse, l'insegnamento si pone un duplice obiettivo: da una parte, fornire la conoscenza di base necessaria per il disegno corretto di interazioni uomo-macchina, che sta alla base della progettazione di applicazioni di ogni genere (web e non, mobili, etc.). Dall'altra, concentrandosi sulle applicazioni web e mobili, che sono oggetto specifico dell'insegnamento, fornire la conoscenza di base necessaria per la progettazione e lo sviluppo di applicazioni Web interattive, accessibili da terminali desktop e mobili (come per esempio gli smart phone), e caratterizzate da una logica applicativa mediamente complessa.

Più precisamente, per quanto riguarda l'interazione uomo-macchina, verranno acquisite competenze sia teoriche sia pratiche nel disegno corretto di interazioni, con specifico riferimento alla progettazione user-centered. Per quanto riguarda gli aspetti architetturali e tecnologici, l'insegnamento tratterà dal punto di vista sia teorico che pratico: (i) lo sviluppo di pagine web statiche e dinamiche responsive; (ii) la programmazione lato server; (iii) la programmazione lato client per device mobili su piattaforma Android e lo sviluppo di interfacce mobili per applicazioni web. Inoltre, l'insegnamento tratterà la rappresentazione e interpretazione di dati con XML, data la sua importanza per l'interoperabilità tra applicazioni.

Per permettere di sperimentare le nozioni apprese durante le ore di teoria in aula, l'insegnamento

include una sostanziale parte di laboratorio. I temi introdotti durante il laboratorio corredano e integrano le conoscenze derivanti dalla parte teorica (knowledge and understanding) e permettono agli studenti e alle studentesse di familiarizzare con le metodologie e tecnologie introdotte, anche investigando soluzioni alternative (applying knowledge and understanding). Inoltre durante le ore di laboratorio è previsto lo sviluppo di un'applicazione realistica con interfaccia sia web che mobile. La preparazione e la discussione del progetto di laboratorio sono inoltre volte a stimolare le capacità di organizzare il lavoro in piccoli gruppi, e poi di illustrare verbalmente le soluzioni adottate (communication skills).

3. Risultati dell'apprendimento attesi:

Gli studenti e le studentesse acquisiranno conoscenza su:

- progettazione user-centered, la prototipazione, la valutazione dell'usabilità e il disegno inclusivo per categorie speciali di utenti.
- concetti rilevanti per analizzare l'interazione: affordance, modelli concettuali, metafore, feedback, etc.
- architetture di riferimento per lo sviluppo di applicazioni web e mobile
- modelli più usati per gestire applicazioni interattive di media complessita`
- tecnologie attualmente utilizzate per l'implementazione delle applicazioni (back-end e interfacce utente web/mobili).

Gli studenti e le studentesse acquisiranno anche la capacita` di sviluppare

- pagine dinamiche PHP
- applicazioni Web a 3 livelli in ambiente Java, utilizzando HTML5-JavaScript, CSS, JSP e Java Servlet, e il framework MVC Angular JS
- applicazioni mobili.

Per quanto riguarda questo ultimo punto, essi/esse saranno in grado di realizzare una applicazione all'interno del sistema Android che interagisca correttamente coi servizi del sistema operativo e le altre applicazioni residenti. Saranno inoltre in grado di saper scrivere un programma usando i costrutti piu' espressivi del linguaggio Python.

4. Modalità di verifica dell'apprendimento:

L'esame è composto da un TEST SCRITTO e da una VERIFICA DI LABORATORIO. Le due prove possono essere sostenute in qualsiasi ordine.

TEST SCRITTO: Prova scritta che include esercizi e domande teoriche sul programma dell'insegnamento. Viene valutata da un minimo di 0 ad un massimo di 30 e si considerano sufficienti i voti >=18. Come da regolamento di Ateneo, ogni studente/studentessa può sostenere un numero massimo di tre prove scritte durante l'Anno Accademico (cioè, consegnare il proprio elaborato tre volte). Il voto ottenuto durante un test scritto decade se si partecipa ad un altro test scritto e si consegna l'elaborato.

VERIFICA DI LABORATORIO: consiste di (1) la discussione di un progetto Web svolto durante il corso, che include una applicazione java con interfaccia Web e mobile Android, e la realizzazione di un programma scritto in linguaggio Python. La discussione deve essere effettuata preferibilmente in unica soluzione, con tutti i membri del gruppo di laboratorio presenti. Il voto di laboratorio è un numero intero compreso tra 0 e 30, si considerano sufficienti i voti >=18.

CALCOLO DEL VOTO FINALE DI ESAME: Sia X il voto del test scritto; sia Y il voto di laboratorio. Il voto Fin finale dell'esame si ottiene come segue: Fin = (X+Y)/2

Note: i voti acquisiti durante la prova di laboratorio, o durante il test scritto, rimangono validi fino al termine della terza sessione d'esame (quella che precede l'inizio del nuovo corso). Quando si superano entrambe le prove, è necessario registrare il voto finale entro i limiti imposti dal Regolamento di Ateneo.

5. Modalità d'insegnamento:

L'insegnamento è diviso in una parte di teoria e una di laboratorio. Per la parte di teoria sono

previste 60 ore di lezione frontali che seguono il programma dell'insegnamento, integrate da casi di studio e da esercitazioni volte ad illustrare l’applicazione pratica dei concetti appena studiati.

La parte di laboratorio è articolata in due moduli da 30 ore ciascuno. Il primo modulo ha per oggetto l’insegnamento di tecnologie per lo sviluppo di pagine web statiche e dinamiche, e di applicazioni server-side MVC con accesso a basi di dati relazionali. Il secondo modulo è suddiviso in due parti: programmazione client-side in ambiente Android utilizzando il linguaggio Java; scrittura di un programma in Python. Le esercitazioni in laboratorio sono svolte a calcolatore, utilizzando ambienti di sviluppo professionali, come Eclipse e/o NetBeans.

Le sperimentazioni che vengono effettuate durante le ore di laboratorio, strutturate come sequenze di esercizi specifici, sono fondamentali per aiutare gli studenti e le studentesse a comprendere e assimilare i contenuti teorici spiegati a lezione in quanto permettono di mettere in pratica i concetti e le metodologie illustrate su esempi concreti. Inoltre lo sviluppo del progetto di laboratorio, che si effettua in gruppo (max 3 persone per gruppo), permette di consolidare le conoscenze teoriche e il lavoro in team in un caso realistico di media complessità.

Si consiglia caldamente la frequenza costante alle lezioni di teoria e di laboratorio. È inoltre fondamentale iscriversi al corso online su I-Learn, all’interno del quale e' messo a disposizione materiale didattico di supporto.

6. Attività di supporto:

Vengono forniti on-line i lucidi delle lezioni, link a documentazione disponibile sul Web ed alcuni testi degli esami scritti degli anni precedenti.

Il materiale didattico di supporto (lucidi delle lezioni, link a documentazione, esempi di testo di esami ed altro) è disponibile presso il supporto on-line ai corsi I-learn (vd. le pagine dei corsi (i) Interazione Uomo Macchina e (ii) Interazione Uomo Macchina e Tecnologie Web - parte tecnologica).

Si noti che i lucidi non sostituiscono il libro di testo, ne' il materiale integrativo ad esso affiancato.

7. Programma:

PARTE 1A: HUMAN-COMPUTER INTERACTION (HCI)

- Human-computer interaction (HCI): Definizioni e contesto, evoluzione di HCI, nuove direzioni.
- Il fattore umano: percezione (gestalt e affordance), attenzione e memoria, modelli mentali, metafore, il modello di Shneiderman e il modello di Norman.
- Disegno di interazioni: user-centered design, requisiti funzionali e di usabilità (raccolta, analisi, presentazione), prototipazione, linee guida (con gestione degli errori ed assistenza agli utenti), elementi di tipografia elettronica, di layout e gestione del colore.
- Tecniche di valutazione: valutazione senza utenti (quantitativa e qualitativa), valutazione con utenti, problemi, presentazione dei risultati.
- Disegno inclusivo: accessibilità, disegno per utenti di differenti gruppi di età (bambini/e, persone anziane), internazionalizzazione.

PARTE 1B - PROGRAMMAZIONE DI DEVICE MOBILI

- Il sistema operativo Android e le sue peculiarita'. Gli elementi costitutivi di una applicazione utente ed il suo ciclio di vita. Le pratiche consolidate e l'ambiente di sviluppo di una applicazione Android.
- Il linguaggio avanzato Python: principali differenze rispetto al linguaggio Java e le sue caratteristiche piu' interessanti.

PARTE 2 - APPLICAZIONI WEB E TECNOLOGIE DI SUPPORTO

- Architetture delle applicazioni Web: Web browser e Web server.
- Il Pattern Model View Controller per le applicazioni Web - Progettazione e sviluppo di applicazioni Web a 3 livelli basate su MVC:
    - Il primo livello (client dell'applicazione): HTML5, CSS, scripting lato client

(JavaScript e AJAX), Java Server Pages (JSP). Raccolta dati con HTML form.
    - Il secondo livello (logica applicativa): Servlet Java.
    - Il terzo livello (livello dei dati): accesso a database relazionali con Java Database Connectivity (JDBC), rappresentazione e gestione di informazioni con XML (XML Schema, XPath, XML Parsers).
- Framework MVC Angular JS per le applicazioni MVC.
- Introduzione al linguaggio PHP per lo sviluppo di pagine web dinamiche con accesso a DB relazionale.

8. Testi consigliati e bibliografia:

PARTE 1A:

- Polillo, R. - FACILE DA USARE, Edizioni Apogeo, 2010

PARTE 1B:

- Android: http://www.html.it/guide/guida-android/
- Python: http://www.python.it/doc/newbie/

PARTE 2:

- Programmazione Web lato Server, di V. Della Mea, L. Di Gaspero, I. Scagnetto, Edizione 2, Apogeo, 2011

ALTRI RIFERIMENTI (per consultazione):

- La caffettiera del masochista, di D. Norman, Apogeo
- Programmazione Java - tecniche avanzate, di Deitel e Deitel. Ed. Pearson - Prentice Hall
- Principi di Web design, di Joe Sklar, Apogeo

Insegnamento	**MFN0590 - Lingua Inglese I**

Insegnamento (inglese):	**English I**
CFU:	**3**
Settore:	**L-LIN/12 - LINGUA E TRADUZIONE - LINGUA INGLESE**
Periodo didattico:	**1** **2**
Tipologia di Attività Formativa:	**E - prova finale e lingua straniera**
Docenti:	**Enrico BINI (Titolare)** **Viviana BONO (Titolare)** **Francesca CORDERO (Titolare)** **Jeanne Marie GRIFFIN (Esercitatore)** **Viviana PATTI (Titolare)** **Daniele Paolo RADICIONI (Titolare)** **Maddalena ZACCHI (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Nessuna.

2. Obiettivi formativi:

Corso di base di Inglese orientato alla comprensione dei testi a alla grammatica di base.

3. Risultati dell'apprendimento attesi:

Al termine dell'insegnamento lo studente deve essere in grado di comprendere e produrre brevi frasi in lingua inglese e leggere semplici testi di argomento scientifico-informatico.

4. Modalità di verifica dell'apprendimento:

Il test finale e` svolto nei laboratori informatici mediante il sistema SET. La prova di Inglese I per informatica e` articolata in due parti, A e B. La parte A (durata 60 min.) deve essere necessariamente superata per poter accedere alla parte B. Il superamento della parte B (durata 30 min.) completa la prova. La valutazione e` automatica e comunicata immediatamente dal sistema allo studente. In ogni appello si svolgono entrambe le parti, in successione; gli studenti che hanno gia` superato la prima parte in un appello precedente devono dare solo la seconda.

Attenzione: Gli studenti possono richiedere il riconoscimento dei certificati B1-B2 (secondo il Common European Framework) per l'esame di Lingua Inglese I compilando l'apposito modulo per il riconoscimento delle APU - Attività Pre-Universitarie (http://di.unito.it/aputarm). Il modulo va consegnato on-line sul proprio spazio MyUniTO oppure alla Segreteria Studenti del Polo delle Scienze della Natura (http://di.unito.it/segreteriastudenti), in via S. Croce 6, secondo le scadenze definite dalla Segreteria, di norma a metà giugno di ogni anno. A seguito di apposita delibera, gli esami riconosciuti saranno direttamente caricati sulla carriera degli studenti e lo studente non dovrà più sostenere Lingua Inglese I – mfn0590 – 3 CFU, facente parte dei crediti obbligatori del piano carriera del I anno.

5. Modalità d'insegnamento:

Le lezioni sono svolte in aula dall'esperto linguistico (Dott.ssa Jeanne Griffin).

6. Attività di supporto:

Sono disponibili dispense redatte dall'esperto linguistico.

Le dispense aggiornate sono disponibili anche su I-learn.

7. Programma:

- Grammatica di base
- Lessico
- Pratica

Informazioni dettagliate su I-learn.

8. Testi consigliati e bibliografia:

- New English File Elementary, Oxford University Press
- Essential English Grammar in Use (B1), by Raymond Murphy and Lelio Pallini
- English Grammar in Use (B2), by Raymond Murphy
- Discussions A-Z, Cambridge University Press

Insegnamento	**MFN1354 - Linguaggi e Paradigmi di Programmazione**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**D - libera**
Docenti:	**Viviana BONO (Titolare)** **Luca PADOVANI (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Conoscenza delle basi della matematica discreta e della programmazione.

Eventuali corsi propedeutici

Matematica discreta e logica; Programmazione 1 e 2.

2. Obiettivi formativi:

I linguaggi di programmazione moderni supportano molteplici paradigmi di programmazione, quello imperativo e quello orientato agli oggetti in primis ma, recentemente, anche quello funzionale. L'approccio funzionale consente al programmatore di lavorare a un più alto livello di astrazione, favorendo lo sviluppo rapido di prototipi con provabili garanzie di correttezza, la modularità e il riuso del codice. L'insegnamento si propone di fornire allo studente una introduzione al paradigma di programmazione funzionale, ai sistemi di tipi per i linguaggi funzionali con i relativi algoritmi di inferenza e alla dimostrazione di correttezza di programmi funzionali. Nella parte avanzata del corso si introdurranno tecniche di programmazione e dimostrazione per strutture dati co-induttive - in particolare gli stream - e si illustrerà l'applicazione delle tecniche apprese a Java il quale, a partire dalla versione 8, ha incorporato costrutti tipici della programmazione funzionale, in particolare le lambda espressioni.

3. Risultati dell'apprendimento attesi:

Al termine dell'insegnamento lo studente dovrà dimostrare di avere compreso i principi alla base della programmazione funzionale. Dovrà quindi dimostrare di conoscere le principali tecniche di programmazione e dimostrazione utilizzate in tale paradigma e le strutture di dati impiegate (in particolare, le liste e gli stream), utilizzandole per impostare e risolvere problemi di programmazione.

Relativamente alla parte avanzata del corso, lo studente dovrà dimostrare di aver appreso la tecnica di dimostrazione per co-induzione e di saper applicare costrutti e pattern tipici della programmazione funzionale al linguaggio Java.

4. Modalità di verifica dell'apprendimento:

Esame scritto in laboratorio con esercizi su parte teorica e laboratorio, in quest'ultimo caso da svolgere al calcolatore.

5. Modalità d'insegnamento:

Lezioni ed esercitazioni in aula ed esercitazioni in laboratorio, in entrambi i casi con sessioni interattive sia su carta che al computer.

6. Attività di supporto:

Non previste.

7. Programma:

Nel seguente elenco di argomenti non viene fatta distinzione tra argomenti svolti in aula ed argomenti svolti in laboratorio, che sono peraltro strettamente connessi.

- Storia dei linguaggi di programmazione, con particolare riferimento a quelli funzionali;
- Calcolo come riscrittura: le basi dell'esecuzione dei programmi funzionali;
- Espressioni e loro tipi. Tipi di base;
- Progettazione di programmi funzionali. Tecniche di ricorsione;
- Liste e funzioni del prim'ordine su liste;
- Dimostrazioni di correttezza di programmi funzionali usando l'induzione;
- Testing di programmi Haskell con QuickCheck;
- L'idea di astrazione funzionale. Funzioni di ordine superiore e pattern di trasformazione;
- Alberi e tipi algebrici generali.

Per la versione da 9 CFU, i seguenti argomenti addizionali:

- Cenni a monadi e programmazione imperativa;
- Strutture di dati infinite (stream) e dimostrazioni di correttezza usando la co-induzione;
- Programmazione funzionale in Java 8.

8. Testi consigliati e bibliografia:

- Simon Thompson, "Haskell: the Craft of Functional Programming" (terza edizione), Addison-Wesley, 2011.
- Raoul Gabriel Urma, Mario Fusco, Alan Mycroft, "Java 8 In Action: Lambdas Streams And Functional Style Programming", Manning Publications, 2014.

Può anche essere utile la consultazione di:

- Graham Hutton, "Programming in Haskell" (seconda edizione), Cambridge University Press, 2016.
- Richard Bird, "Thinking Functionally with Haskell", Cambridge University Press, 2014.

Insegnamento

MFN0610 - Linguaggi e Paradigmi di Programmazione

Insegnamento (inglese):	
CFU:	**9**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Viviana BONO (Titolare)** **Luca PADOVANI (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Conoscenza delle basi della matematica discreta e della programmazione.

Eventuali corsi propedeutici

Matematica discreta e logica; Programmazione 1 e 2.

2. Obiettivi formativi:

I linguaggi di programmazione moderni supportano molteplici paradigmi di programmazione, quello imperativo e quello orientato agli oggetti in primis ma, recentemente, anche quello funzionale. L'approccio funzionale consente al programmatore di lavorare a un più alto livello di astrazione, favorendo lo sviluppo rapido di prototipi con provabili garanzie di correttezza, la modularità e il riuso del codice. L'insegnamento si propone di fornire allo studente una introduzione al paradigma di programmazione funzionale, ai sistemi di tipi per i linguaggi funzionali con i relativi algoritmi di inferenza e alla dimostrazione di correttezza di programmi funzionali. Nella parte avanzata del corso si introdurranno tecniche di programmazione e dimostrazione per strutture dati co-induttive - in particolare gli stream - e si illustrerà l'applicazione delle tecniche apprese a Java il quale, a partire dalla versione 8, ha incorporato costrutti tipici della programmazione funzionale, in particolare le lambda espressioni.

3. Risultati dell'apprendimento attesi:

Al termine dell'insegnamento lo studente dovrà dimostrare di avere compreso i principi alla base della programmazione funzionale. Dovrà quindi dimostrare di conoscere le principali tecniche di programmazione e dimostrazione utilizzate in tale paradigma e le strutture di dati impiegate (in particolare, le liste e gli stream), utilizzandole per impostare e risolvere problemi di programmazione.

Relativamente alla parte avanzata del corso, lo studente dovrà dimostrare di aver appreso la tecnica di dimostrazione per co-induzione e di saper applicare costrutti e pattern tipici della programmazione funzionale al linguaggio Java.

4. Modalità di verifica dell'apprendimento:

Esame scritto in laboratorio con esercizi su parte teorica e laboratorio, in quest'ultimo caso da svolgere al calcolatore. Rispetto alla versione del corso da 6 cfu, la prova di esame comprende esercizi relativi agli argomenti aggiuntivi (stream, programmazione funzionale in Java 8).

5. Modalità d'insegnamento:

Lezioni ed esercitazioni in aula ed esercitazioni in laboratorio, in entrambi i casi con sessioni interattive sia su carta che al computer.

6. Attività di supporto:

Non previste.

7. Programma:

Nel seguente elenco di argomenti non viene fatta distinzione tra argomenti svolti in aula ed argomenti svolti in laboratorio, che sono peraltro strettamente connessi.

- Storia dei linguaggi di programmazione, con particolare riferimento a quelli funzionali;
- Calcolo come riscrittura: le basi dell'esecuzione dei programmi funzionali;
- Espressioni e loro tipi. Tipi di base;
- Progettazione di programmi funzionali. Tecniche di ricorsione;
- Liste e funzioni del prim'ordine su liste;
- Dimostrazioni di correttezza di programmi funzionali usando l'induzione;
- Testing di programmi Haskell con QuickCheck;
- L'idea di astrazione funzionale. Funzioni di ordine superiore e pattern di trasformazione;
- Alberi e tipi algebrici generali.

Per la versione da 9 CFU, i seguenti argomenti addizionali:

- Cenni a monadi e programmazione imperativa;
- Strutture di dati infinite (stream) e dimostrazioni di correttezza usando la co-induzione;
- Programmazione funzionale in Java 8.

8. Testi consigliati e bibliografia:

- Simon Thompson, "Haskell: the Craft of Functional Programming" (terza edizione), Addison-Wesley, 2011.
- Raoul Gabriel Urma, Mario Fusco, Alan Mycroft, "Java 8 In Action: Lambdas Streams And Functional Style Programming", Manning Publications, 2014.

Può anche essere utile la consultazione di:

- Graham Hutton, "Programming in Haskell" (seconda edizione), Cambridge University Press, 2016.
- Richard Bird, "Thinking Functionally with Haskell", Cambridge University Press, 2014.

Insegnamento **MFN0603 - Linguaggi Formali e Traduttori**

Insegnamento (inglese):	
CFU:	**9**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Mario COPPO (Titolare)** **Viviana PATTI (Titolare)** **Jeremy James SPROSTON (Titolare)** **Maddalena ZACCHI (Professore a Contratto)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Lo studente deve avere familiarità con i concetti fondamentali della logica, della teoria degli insiemi e della progettazione di algoritmi iterativi e ricorsivi. Deve inoltre aver acquisito capacità di programmare in linguaggi ad alto livello.

Eventuali corsi propedeutici

Le competenze richieste per una proficua frequenza delle lezioni sono fornite dagli insegnamenti: Programmazione I e laboratorio, Programmazione II e laboratorio, Architettura degli elaboratori, Matematica Discreta e Logica.

2. Obiettivi formativi:

Conoscenze nel campo della descrizione formale dei linguaggi e della traduzione (in particolare della compilazione) sono sempre state considerate fondamentali nel bagaglio culturale di un informatico e non possono essere ignorate dagli addetti al settore. Competenze di questo tipo si trovano nei curricula di orientamento informatico-matematico delle Università di tutto il mondo. I linguaggi di programmazione si sono evoluti presentando nuovi problemi di compilazione che hanno portato allo sviluppo di metodi generali per affrontarli. Buona parte della tecnologia di “front-end” dei compilatori, come grammatiche, espressioni regolari, parsificatori e traduttori guidati dalla sintassi, trovano anche applicazione in tutti i programmi in cui sia richiesta l’analisi strutturale di un testo o, in generale, di dati in cui si debba individuare una struttura.

L’insegnamento si propone pertanto di fornire allo studente una visione introduttiva dei problemi connessi alla definizione e alla traduzione dei linguaggi di programmazione, con particolare riferimento al progetto e alla costruzione di compilatori. Le metodologie e le tecniche presentate sono utili in generale come formalismi per definire il comportamento di un sistema o per realizzare componenti che richiedono una traduzione tra rappresentazioni diverse di dati. Scopo del laboratorio è quello di completare la preparazione fornendo competenze pratiche che riguardano l’applicazione dei concetti presentati nelle lezioni di teoria. In particolare l’obiettivo del laboratorio è quello di realizzare un semplice compilatore applicando le tecniche corrette.

3. Risultati dell'apprendimento attesi:

Comprensione e acquisizione della terminologia tecnica del settore. Conoscenza delle metodologie fondamentali per la descrizione della sintassi di linguaggi formali (ad esempio i linguaggi di programmazione) e delle principali tecniche di parsificazione e traduzione; capacità di utilizzare tali conoscenze per lo sviluppo di sistemi software. Conoscenza e padronanza degli strumenti di base per la progettazione di traduttori.

4. Modalità di verifica dell'apprendimento:

Ai fini della determinazione del voto finale l'esame dell’insegnamento di Linguaggi Formali e Traduttori è diviso in due parti. 1. Scritto. Lo scritto è formato da: a) domande di teoria su tutto il programma svolto a lezione; b) esercizi che intendono verificare che lo studente sia in grado di applicare quanto appreso.

2. Discussione sull'esercitazione finale di laboratorio consistente nella realizzazione di un interprete/compilatore per un semplice linguaggio definito ad hoc.

Per superare l’esame lo studente deve raggiungere la sufficienza in entrambe le parti. Per sostenere la prova di laboratorio è necessario aver superato la prova scritta. Entrambe le prove devono essere superate nella stessa sessione d'esame. A richiesta dello studente, e comunque per ottenere la lode, è possibile sostenere una prova orale integrativa.

5. Modalità d'insegnamento:

L’insegnamento è suddiviso in una parte di teoria e una di laboratorio.

Per la parte di teoria sono previste 48 ore di lezioni frontali. Nel corso delle lezioni vengono anche svolti esercizi esemplificativi. Agli studenti sono resi disponibili le slide eventualmente usate dal docente come supporto alle lezioni.

La parte di laboratorio consiste in un modulo di 30 ore. Agli studenti vengono presentati una serie di esercizi relativi all'applicazione pratica dei concetti trattati nella parte di teoria. Gli esercizi vengono svolti in laboratorio con l'assistenza del docente. I primi esercizi avranno lo scopo di fornire agli studenti l'esperienza necessaria per affrontare l'esercitazione finale, che richiede la realizzazione di un interprete/compilatore.

6. Attività di supporto:

Le lezioni frontali vengono integrate da 12 ore non creditizzate, con lo scopo di svolgere esercizi e di rispondere alle eventuali richieste da parte degli studenti di approfondimento di argomenti del programma.

Il materiale didattico di supporto (programma dettagliato con riferimento ai capitoli dei libri, slide, esempi di testi d'esame ed altro) sarà reso disponibile sulla piattaforma I-learn.

7. Programma:

Automi a stati finiti ed espressioni regolari, analisi lessicale. Grammatiche e famiglie di linguaggi. Analisi sintattica top-down e bottom-up. Traduzione diretta dalla sintassi. Generazione del codice intermedio.

Laboratorio: esercizi di programmazione che riguardano, in particolare, l'analisi lessicale (implementazione di automi), l'analisi sintattica (realizzazione di analizzatori a discesa ricorsiva), e la generazione di codice intermedio (sviluppo di un compilatore per un semplice linguaggio).

Il programma dettagliato dell'insegnamento sarà pubblicato sulla piattaforma I-learn

8. Testi consigliati e bibliografia:

A. V. Aho, M. S. Lam, R. Sethi, J.D. Ullman, "Compilatori: Principi, tecniche e strumenti", Pearson Paravia Bruno Mondadori S.p.A., 2009, ISBN 978-88-7192-559-2.

John E. Hopcroft, R. Motwani, J. D. Ullman, “Automi, Linguaggi e Calcolabilità”, Pearson-Addison Wesley, 2009, ISBN 978-88-7192-552-3.

Per la parte di laboratorio, agli studenti viene fornito, oltre ai testi degli esercizi, materiale utile per il loro svolgimento.

Insegnamento **INF0003 - Logica per l'Informatica**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**MAT/01 - LOGICA MATEMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**D - libera**
Docenti:	**Simonetta RONCHI DELLA ROCCA (Professore a Contratto)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

- Conoscenza di base di Logica - Elementi di linguaggi funzionali.

Eventuali corsi propedeutici

- Logica e Matematica Discreta - Linguaggi e Paradigmi di Programmazione

2. Obiettivi formativi:

Si vogliono fornire agli studenti strumenti formali per il ragionamento rigoroso sia in linguaggio naturale che in linguaggi formali. Si vuole utilizzare la logica come strumento per una descrizione rigorosa della specifica di un programma e per porre le basi della verifica automatica di proprietà di programmi.

3. Risultati dell'apprendimento attesi:

Lo studente dovrà dimostrare di aver appreso i principi fondamentali della logica formale Dovrà inoltre dimostrare di di conoscere la relazione della logica formale con la programmazione, sia funzionale che logica.

4. Modalità di verifica dell'apprendimento:

Esame orale.

5. Modalità d'insegnamento:

Lezioni alla lavagna.

6. Attività di supporto:

Non previste

7. Programma:

- Logica proposizionale - Sistemi deduttivi: deduzione naturale, correttezza e completezza - Logica proposizionale e linguaggi di programmazione: isomorfismo di Curry-Howard - Logica dei predicati - Calcolo del primo ordine: deduzione naturale, correttezza e completezza - Metodo di risoluzione.

8. Testi consigliati e bibliografia:

Asperti A., Ciabattoni A., "Logica ad Informatica", Mc Graw-Hill Education, 1996. (parti scelte)

Goulbault-Larrecq J., Makie I., "Proof Theory and Automated Deduction", Kluwer Academic Publisher. (parti scelte)

Materiale a cura del docente sara' introdotto nella pagina moodle del corso.

Insegnamento **MFN0578 - Matematica Discreta e logica**

Insegnamento (inglese):	**Discrete Mathematics and Logic**
CFU:	**12**
Settore:	**MAT/01 - LOGICA MATEMATICA** **MAT/02 - ALGEBRA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**A - di base**
Docenti:	**Alessandro ARDIZZONI (Titolare)** **Andrea MORI (Titolare)** **Luca MOTTO ROS (Titolare)** **Matteo VIALE (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Conoscenza delle basi della matematica della scuola superiore: in particolare le operazioni aritmetiche di base, le proprietà delle potenze, le equazioni di primo e secondo grado. Conoscenza della terminologia di base relativa alle parti del discorso: nomi, verbi, proposizioni, aggettivi.

Eventuali corsi propedeutici

Nessuno

2. Obiettivi formativi:

L’insegnamento si propone di fornire allo studente una introduzione alla matematica discreta e alla logica matematica, con particolare riguardo per i loro aspetti piu` rilevanti per la formazione di base di un informatico, in particolare una adeguata familiarita` con le strutture algebriche, il calcolo combinatorio e le principali tecniche di dimostrazione.

3. Risultati dell'apprendimento attesi:

MATEMATICA DISCRETA. Al termine del corso lo studente dovrà da prova di aver compreso le basi fondamentali della teoria della teoria dei gruppi e degli anelli e dei loro morfismi, con la conoscenza di alcuni specifici esempi; in particolare ci si aspetta la conoscenza della struttura algebrica degli interi, con applicazione all’aritmetica modulare. Lo studente dovrà essere in grado di manipolare permutazioni, risolvere equazioni diofantee lineari in due variabili, congruenze lineari ed impostare e risolvere problemi di carattere enumerativo utilizzando in modo appropriato il linguaggio ed il formalismo della teoria degli insiemi, com particolare riguardo alle operazioni tra insiemi (unione, intersezione, prodotto), le relazioni e le funzioni.

LOGICA Al termine del corso lo studente dovra` dimostrare di sapere riconoscere le principali tecniche di dimostrazione (diretta, per assurdo, per contrapposizione, per casi) in semplici dimostrazioni di proposizioni relative a strutture algebriche e relazionali. Dovra` essere in grado di utilizzare le principali forme del principio di induzione (ordinaria, forte, principio del minimo) in semplici dimostrazioni aritmetiche o relative alla sintassi formale dei linguaggi proposizionali e del prim’ordine. Dovra` dimostrare di essere in grado di formalizzare mediante formule logiche semplici asserzioni formulate in italiano. Dovrà essere in grado di valutare la validità o meno di semplici formule del primo ordine in strutture algebriche in accordo con le regole della semantica di Tarski e dovrà essere in grado di stabilire quando una formula non è conseguenza logica di altre formule provvedendo semplici strutture algebriche o relazionali che producono un controesempio. Dovra` inoltre dimostrare di conoscere la terminologia di base della teoria dei reticoli e delle algebre di Boole, con particolare attenzione ai concetti di atomo, insieme inferiore, distributività, assieme al loro utilizzo nella rappresentazione dei reticoli distributivi e delle algebre di Boole finite. Infine, lo studente dovra` dimostrare di riconoscere l’esistenza o meno di biiezioni tra insiemi

infiniti costruiti mediante le operazioni di unione disgiunta, prodotto cartesiano, insieme potenza, e insieme delle sequenze finite.

ENGLISH

DISCRETE MATHEMATICS At the end of the course the student will have to show an understanding of the basic structure of groups and rings and their morphisms, with knowledge of some specific examples; in particular the student is expected to know the algebraic structure of the integers, with applications to modular arithmetic. The student will have to be able to manipulate permutations, solve linear diophantine equations in two variables, linear congruences and set and solve enumerative problems using in a proper way the language and formalism of set theory, specifically using set operations (union, intersection, product), relations and functions.

LOGIC Once the course is completed the student must show a basic understanding of the main proof techniques employed in basic mathematical arguments (direct proof, proof by contraposition, proof by cases) at least for the case of basic arguments establishing elementary properties of algebraic and relational structures. The student must also be able to master the use of the induction principle in simple arguments regarding arithmetic properties, or syntactic properties of propositional or first order languages. The student must also be able to formalize in first order logic simple assertions about arithmetic properties of natural or real numbers. The student must also be able to apply the rules of Tarski semantic in order to establish the truth or falsity of a given formula in a given structure, moreover he must also be able to establish when a given formula is not a logical consequence of other formulae providing simple relational or algebraic structures which give a counterexample. The student must also know the basic terminology for lattices and boolean algebras with a particular emphasis on the notions of distributivity, infimum of a subset of a lattice, and their use in the representation of finite lattices. Finally the student must have a basic familiarity with the set theoretic concepts of function,bijection, surjection, injection and must be able to recognize whether there exist or not a bijection between certain types of infinite sets.

4. Modalità di verifica dell'apprendimento:

ITALIANO L’esame consiste di due parti, una di Matematica Discreta e una di Logica, che possono essere affrontate anche in appelli differenti, tuttavia entro lo stesso anno accademico (dall’appello di Gennaio a quello di Settembre). Le prove di esame sono scritte, a meno di motivata richiesta di esame orale da parte dei docenti. Le domande possono riguardare sia la teoria che lo svolgimento di esercizi. Dopo il superamento di una delle due parti, non saranno consentiti piu` di due tentativi per l’altra parte, pena l’annullamento degli esiti positivi gia` conseguiti. L’esito dell’esame e` espresso in trentesimi come media dei punteggi riportati nelle parti di Matematica Discreta e di Logica.

ENGLISH. The exam consists of two independent parts, Discrete Mathematics and Logic, that can be taken in different sessions but always within the same Academic Year (from January to September). The two parts of the exams are written, unless the teachers require an oral exam. The questions in the exams can either be theoretical or problem-solving in nature. From the moment that one of the two parts of the exams has been successfully taken the student will have at most two attempts to complete the other part, after which the first part taken will be voided. The final grade of the exam is obtained as the average of the two parts, expressed as a fraction of 30.

5. Modalità d'insegnamento:

ITALIANO Lezioni frontali (52 ore Matematica Discreta e 52 ore Logica) comprese le esercitazioni in aula.

ENGLISH Lectures (52 hrs Discrete Mathematics + 52 hrs Logic) including exercises in classroom.

6. Attività di supporto:

ITALIANO Disponibilita` di un tutor.

ENGLISH A tutor will be available.

7. Programma:

Linguaggio degli insiemi (14 ore circa) • Insiemi: insieme vuoto; sottoinsiemi; unione; intersezione; complementare; insieme delle parti (con particolare attenzione al caso finito). • Corrispondenze, relazioni e funzioni: relazioni d'ordine. • Relazioni di equivalenza e partizioni. • Composizione e inversione di corrispondenze. • Iniettivita`, suriettivita`, composizione e invertibilita` di funzioni.

Calcolo combinatorio (12 ore circa) • Cardinalita` di insiemi finiti • Principi della somma e del prodotto • Disposizioni semplici e con ripetizioni • Combinazioni semplici e con ripetizioni. • Il Teorema del binomio e il triangolo di Pascal-Tartaglia • Il principio di inclusione-esclusione

Strutture algebriche (10 ore circa) • Semigruppi e loro morfismi. • Monoidi e loro morfismi: monoide delle parole • Gruppi e loro morfismi. • Alcuni esempi di strutture algebriche: numeri naturali e interi, gruppo delle biiezioni di un insieme. • Gruppi e sottogruppi ciclici. • Sottogruppi e Teorema di Lagrange. • Anelli: anello delle Matrici. • Corpi e campi: campo dei numeri razionali.

Aritmetica modulare (8 ore circa) • Anelli degli interi e delle classi di resto. • Teorema della divisione • L'algoritmo di Euclide • Identità di Bezout • Equazioni diofantee • Il teorema di Eulero-Fermat

Gruppo delle permutazioni (8 ore circa) • Composizione, potenze e inverse di permutazioni. • Decomposizione in cicli disgiunti e decomposizione in trasposizioni. • Parita` di una permutazione • Sottogruppi del gruppo delle permutazioni.

Logica

Tecniche di dimostrazione (8 ore circa) • Dimostrazione diretta, per assurdo, per contrapposizione; • Connettivi logici e loro significato in termini di condizioni di verita`; • Tavole di verita` e conseguenza logica tra proposizioni.

Il principio di induzione (6 ore circa) • Forma ordinaria e forte del principio di induzione; • Principio del minimo; • Equivalenza tra forme del principio di induzione; • Induzione strutturale; • Ricorsione.

Formalizzazione (8 ore circa) • Linguaggi proposizionali e del prim'ordine: termini, quantificatori, alfabeto non logico, formule; • Schemi di traduzione da linguaggio naturale in linguaggi del prim'ordine (condizione sufficiente, necessaria, per tutti gli n abbastanza grandi, ci sono n arbitrariamente grandi,...)

Semantica della Logica del primo ordine (14 ore circa): Regole della semantica di Tarski per formule del primo ordine: esempi, e definizioni — Conseguenza logica tra formule del primo ordine: definizioni e costruzione di contro-esempi

Reticoli (6 ore circa) • Ordinamenti parziali, con esempi; • Minimo confine superiore, massimo confine inferiore: reticoli; • Distributivita`;

Algebre di Boole (6 ore circa) • Algebra degli insiemi; • Algebre di Boole: definizioni; • Algebre di Boole come reticoli; • Cenni alla rappresentazione delle algebre di Boole finite: atomi; • Calcolo delle funzioni booleane e algebra della commutazione; • Algebra della logica: algebre di Boole e calcolo proposizionale.

Insiemi infiniti (4 ore circa) • Insiemi numerabili e piu` che numerabili: esempi; • Operazioni infinitarie: unioni e intersezioni, prodotti e somme, con le principali proprieta`.

8. Testi consigliati e bibliografia:

A. Mori, Lezioni di Matematica Discreta.

A. Facchini, *Algebra e matematica discreta, per studenti di informatica, ingegneria, fisica e matematica con numerosi esempi ed esercizi svolti*. Zanichelli/Decibel, 2000.

C. Bertone, M. Roggero, *Appunti ed Esercizi di Matematica Discreta*, 2017/18.

F. Preparata, R.T. Yeh, *Introduzione alle strutture discrete*, Boringhieri, 1976.

Andretta, Cardone, Motto Ros, Viale: *Dispense di logica matematica*, 2017.

Insegnamento **MFN0633 - Metodi Formali dell'Informatica**

Insegnamento (inglese):	
CFU:	**9**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1** **2**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Ugo DE' LIGUORO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Sono richieste conoscenze elementari di logica e buone conoscenze di programmazione e algoritmi; inoltre si assume che lo studente possegga nozioni di linguaggi formali.

Eventuali corsi propedeutici

Matematica Discreta e Logica, Programmazione 1 e 2, Algoritmi e Strutture Dati, Linguaggi Formali e Traduttori.

2. Obiettivi formativi:

Il corso si propone di offrire agli studenti le conoscenze di base relative alla verifica formale di proprietà dei programmi. A questo scopo vengono introdotte le basi teoriche della logica dei programmi, il linguaggio JML di specifica per programmi Java e la sua interpretazione nella logica dinamica. Inoltre saranno introdotte tecniche di analisi statica di programmi con puntatori e la separation logic su cui si basano tool come infer.

Il corso è basato sull'uso del framework KeY (libero ed open-source) per la verifica di programmi industriali assistita da calcolatore.

3. Risultati dell'apprendimento attesi:

Conoscenza di alcune tecniche per la verifica formale di programmi basate sulla logica e sulla specifica mediante "contratti". Abilità nel costruire dimostrazioni formali di correttezza dei programmi mediante l'uso di un tool. Consapevolezza della necessità di sviluppare il codice in modo ben strutturato e di documentarlo in modo formalmente preciso per facilitarne la verifica ed assicurarne l'affidabilità.

4. Modalità di verifica dell'apprendimento:

L'esame consiste in un esame orale, in cui saranno verificate non solo le conoscenze teoriche, ma anche le abilità pratiche nell'uso dei tool di verifica interattiva utilizzati durante il corso.

5. Modalità d'insegnamento:

Le lezioni si svolgono in aula sia in modo tradizionale che interattivo mediante esercitazioni. Si farà ampio uso della piattaforma Moodle per la proposta di esercizi e per la discussione del loro svolgimento.

6. Attività di supporto:

Il materiale didattico di supporto (a cura del docente) si può reperire nella pagina Moodle del corso stesso.

7. Programma:

- Semantica dei linguaggi di programmazione

** Semantica operazionale strutturale: espressioni aritmetiche e definizioni ricorsive
** Semantica operazionale di un semplice linguaggio imperativo
** Semantica operazionale del nucleo funzionale di Java: Featherweight Java

- Logica dei programmi

** Logica del primo ordine
** Asserzioni e logica di Floyd-Hoare - HL
** Automazione della verifica dei programmi, verification conditions
** Calcolo degli updates e verifica con KeY-Hoare

- Specifica e verifica di programmi Java

** Specifiche formali in Java Modeling Language - JML
** Logica dinamica per Java - JavaDL
** Verifica di programmi Java con KeY

- Analisi dei puntatori e separation logic

** Analisi statica: analisi del flusso e dei dati
** Analisi dei puntatori
** Separation logic

8. Testi consigliati e bibliografia:

- W. Ahrendt et alii: "Deductive Software Verification - The KeY Book", Springer, 2016.

- G. Winskel: "La semantica formale dei linguaggi di programmazione", The MIT Press, edizione italiana UTET, 1999 (capitoli scelti).

- R. De Nicola, A. Piperno: "Semantica operazionale e denotazionale dei linguaggi di programmazione", Città studi, 1999 (approfondimenti).

Ulteriori riferimenti ad articoli e dispense del docente saranno forniti attraverso la piattaforma Moodle.

Insegnamento	**INF0193 - Metodologie e Tecnologie Didattiche per l'Informatica (PREFIT)**

Insegnamento (inglese):	**Methodologies and technologies for teaching informatics (PREFIT)**
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**D - libera**
Docenti:	**Giuseppina Barbara DEMO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

2. Obiettivi formativi:

3. Risultati dell'apprendimento attesi:

4. Modalità di verifica dell'apprendimento:

5. Modalità d'insegnamento:

6. Attività di supporto:

7. Programma:

8. Testi consigliati e bibliografia:

Insegnamento **MFN0582 - Programmazione I**

Insegnamento (inglese):	**Programming I**
CFU:	**9**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**A - di base**
Docenti:	**Felice CARDONE (Titolare)** **Valentina GLIOZZI (Titolare)** **Alessandro MAZZEI (Titolare)** **Luca ROVERSI (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Non è richiesto alcun prerequisito specifico alla programmazione. È consigliabile avere capacita' di uso del calcolatore con sistema a finestre. È invece opportuno possedere conoscenze di base, quali i concetti di numero (naturale, intero, razionale, reale), di funzione, le quattro operazioni, elevamento a potenza, radice, esponenziale, logaritmo, piano cartesiano, calcolo algebrico elementare. Infine, è indispensabile una buona padronanza della lingua madre, possibilmente accompagnata da una propensione al ragionamento strutturato.

Eventuali corsi propedeutici

Nessuno.

2. Obiettivi formativi:

Fornire i concetti di base della programmazione strutturata ed imperativa di alto livello, appoggiandosi ad un linguaggio di programmazione di riferimento.

3. Risultati dell'apprendimento attesi:

Capacità di: (i) formalizzare la soluzione a problemi computazionali di base per mezzo di costrutti linguistici iterativi e ricorsivi; (ii) tradurre la formalizzazione al punto (i) in un linguaggio di programmazione di riferimento; (iii) valutare correttezza parziale e terminazione di procedure per mezzo di semplici dimostrazioni formali; (iv) valutare l'efficienza di una procedura; (v) rappresentare istantanee dello stato in cui si trovi la memoria di un calcolatore che interpreti programmi ragionevolmente complessi nel linguaggio di programmazione di riferimento.

4. Modalità di verifica dell'apprendimento:

Sono impostate per verificare conoscenza e comprensione degli argomenti sviluppati nel programma didattico, in accordo con i "Descrittori di Dublino", relativi ai learning outcomes. Per mezzo di un numero congruo di esercizi saranno verificate conoscenze e capacità di:

- progettare algoritmi iterativi e ricorsivi;

codificare programmi che implementano algoritmi iterativi e ricorsivi;

- risolvere esercizi sia su aspetti fondamentali della programmazione di alto livello (correttezza, terminazione e costo degli algoritmi) sia sulla simulazione della gestione della memoria.

Gli esercizi saranno scritti o al calcolatore. La somma dei punteggi degli esercizi proposti permetterà di ottenere voti sino al "30 e Lode".

5. Modalità d'insegnamento:

Le lezioni in aula sono svolte alla lavagna (virtuale o meno) e, se opportuno, con l'ausilio di calcolatore.

Il laboratorio e' organizzato a turni. Il numero di turni puo' variare di anno in anno a seconda del numero di iscritti. E' usuale avere due turni per il corso A e due per il corso B.

Istruzioni su come accedere al laboratorio, saranno comunicate in aula, durante le prime lezioni, e rese disponibili sul sito di supporto alla didattica.

6. Attività di supporto:

Sono previste 12 ore facoltative di esercitazione in aula in cui saranno proposti esercizi e relative soluzioni sugli argomenti principali del corso. On-line saranno disponibili:

- dispense su argomenti fondamentali del corso, non esaurientemente coperti dal testo di riferimento;
- documentazione disponibile su Internet;
- software.

7. Programma:

- Struttura di base di un calcolatore;
- Informazioni essenziali su linguaggi di programmazione, differenze tra compilatori e interpreti;
- Algoritmi iterativi e ricorsivi;
- Linguaggio di riferimento: variabili, tipi di dato fondamentali e array, assegnazione e controllo del flusso, procedure e funzioni con parametri, modello di gestione della memoria;
- Correttezza parziale, terminazione e nozioni di costo degli algoritmi.

8. Testi consigliati e bibliografia:

- Walter Savitch "Programmazione di base e avanzata con Java", Pearson --- ISBN 9788865181904.
- Dispense e lucidi ufficiali, distribuiti dai docenti.

Insegnamento **MFN0585 - Programmazione II**

Insegnamento (inglese):	**Programming II**
CFU:	**9**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**A - di base**
Docenti:	**Stefano BERARDI (Titolare)** **Ferruccio DAMIANI (Titolare)** **Diego MAGRO (Titolare)** **Luca PADOVANI (Titolare)** **Gianluca TORTA (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Agli studenti è richiesta la conoscenza delle nozioni fondamentali di programmazione imperativa nel linguaggio Java (comandi di assegnamento, condizionali e iterativi, tipi di dato semplici e array, nozioni di astrazione procedurale e ricorsione).

Eventuali corsi propedeutici

Programmazione I e Laboratorio

2. Obiettivi formativi:

Il corso si propone di raffinare le capacità di programmare nel linguaggio Java apprese nel corso di Programmazione I e di introdurre le nozioni fondamentali della programmazione orientata agli oggetti. In particolare, il corso illustrerà le *astrazioni fondamentali* per la progettazione del software (classi e oggetti), la definizione di *semplici strutture dati* (liste, alberi, pile, code) e operazioni corrispondenti, i meccanismi di base per favorire *riuso e modularità* del software (ereditarietà, polimorfismo, tipi generici), la specifica degli *invarianti di classe* e gestione delle loro violazioni (asserzioni ed eccezioni), così come alcune *classi fondamentali* della libreria Java. Si darà particolare enfasi agli aspetti di buona progettazione del software, utilizzando concetti presi a prestito dall'ingegneria del software e formalismi grafici quali UML.

3. Risultati dell'apprendimento attesi:

Lo studente acquisirà la capacità di progettare e realizzare piccole/medie applicazioni in Java sfruttando gli strumenti di base della programmazione a oggetti, imparerà a utilizzare ed a realizzare alcune strutture dati fondamentali ed approfondirà la conoscenza della libreria standard di Java.

4. Modalità di verifica dell'apprendimento:

L'esame si compone di una verifica di laboratorio e di una prova scritta.

La *verifica di laboratorio* consiste in una prova pratica di programmazione al calcolatore della durata di 45 minuti, viene valutata con un punteggio da 0 a 2 e considerata sufficiente se tale punteggio è 1 o 2. Una volta superata la verifica di laboratorio, il punteggio ottenuto rimarrà valido fino all'appello precedente l'avvio del corso dell'anno successivo. Uno studente può consegnare al massimo 3 elaborati di laboratorio durante l'anno accademico. Può però presentarsi ad ogni appello (previa registrazione) e decidere se consegnare o meno il proprio elaborato.

La *prova scritta*, che può essere sostenuta solo previo superamento della verifica di laboratorio, consiste in esercizi di programmazione e sulle nozioni fondamentali del corso e viene valutata con

un punteggio da 0 a 30. È considerata sufficiente se tale punteggio è 18 o superiore.

Il voto finale dell'esame di Programmazione II e Laboratorio è pari al punteggio della prova scritta. La lode viene assegnata se tale punteggio è 30 ed il punteggio della prova di laboratorio è 2.

5. Modalità d'insegnamento:

L’insegnamento consta di una parte teorica (60 ore) e una pratica (30 ore).

La parte teorica si svolge in aula con l'ausilio di lavagna e gesso ed eventualmente di un calcolatore usato dal docente per la proiezione di lucidi e/o lo sviluppo di semplici programmi.

La parte pratica si svolge nei laboratori didattici in cui il docente illustra esercizi di programmazione che gli studenti risolvono al calcolatore. Il docente (ed eventualmente studenti tutor) supportano gli studenti nella risoluzione degli esercizi assegnati.

6. Attività di supporto:

Il materiale usato durante le lezioni in aula e in laboratorio, i testi degli esercizi e delle prove d'esame sono documentati e/o resi disponibili sulla pagina moodle del corso http://informatica.i-learn.unito.it/.

7. Programma:

Il programma del corso è basato sul testo di riferimento, eventualmente integrato da dispense e/o altro materiale fornito dal docente e reso disponibile sulla pagina moodle del corso. Segue un sommario degli argomenti trattati e dei capitoli relativi sul testo di riferimento. Tale sommario non riflette necessariamente l’ordine di esposizione degli argomenti.

- ripasso dei concetti di base della programmazione imperativa e di Java (Cap. 1-7)
- incapsulamento, definizione di una classe, creazione e uso di oggetti (Cap. 8 e 9)
- specializzazione di una classe, ereditarietà (Cap. 10)
- relazioni tra classi, polimorfismo, classi parzialmente definite, interfacce (Cap. 11)
- generalizzazione di una classe, classi parametriche, tipi generici (Cap. 12)
- invarianti di classe, asserzioni, eccezioni e loro gestione (Cap. 13)
- progettazione e implementazione di strutture dati (liste, alberi, pile, code) (Cap. 15)
- collezioni e iteratori (Cap. 16)
- cenni su progettazione a oggetti e UML

8. Testi consigliati e bibliografia:

Libro di testo:

- W. Savitch: Programmazione di base e avanzata con Java, Seconda edizione, Pearson, 2018 (anche la prima edizione va bene).

Altri testi di riferimento:

- M. Naftalin, P. Wadler: Java Generics and Collections, O'Reilly, 2006.
- G. Pighizzini, M. Ferrari: Dai fondamenti agli oggetti, Terza edizione, Pearson Addison Wesley, 2008.

Insegnamento **MFN0605 - Programmazione III**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Liliana ARDISSONO (Titolare)** **Marco BOTTA (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Si richiede una buona conoscenza dell'analisi e della progettazione object-oriented (fornita dagli insegnamenti Programmazione II e Algoritmi e strutture dati) e dei meccanismi di base e delle problematiche della programmazione concorrente (fornita dall'insegnamento Sistemi Operativi). Gli studenti e le studentesse devono inoltre avere la capacità di scrivere, compilare e verificare la correttezza di programmi in Java.

Eventuali corsi propedeutici

Programmazione II, Algoritmi e strutture dati, Sistemi Operativi.

2. Obiettivi formativi:

Lo sviluppo di software efficiente e scalabile presuppone la capacità di programmare applicazioni distribuite e concorrenti. In particolare, la programmazione distribuita in ambiente object oriented arricchisce le nozioni di base di programmazione concorrente sfruttando il paradigma ad oggetti per una più chiara scomposizione delle attività da eseguire in parallelo e loro attribuzione alle entità software di competenza, che possono essere modellate come oggetti distribuiti che offrono i relativi servizi.

L'insegnamento si pone l’obiettivo di fornire la conoscenza di base necessaria per la programmazione di applicazioni object-oriented distribuite e concorrenti, usando linguaggi ad alto livello, attraverso (i) l’invocazione remota di metodi degli oggetti, e (ii) la programmazione di thread paralleli, cioè di processi “leggeri” che possano operare su uno o più processori all’interno della stessa applicazione principale. Altro obiettivo fondamentale dell'insegnamento è la tecnica di programmazione ad eventi per la realizzazione di interfacce grafiche, che stanno alla base di tutte le applicazioni desktop e web basate su finestre. Tutte le conoscenze verranno fornite utilizzando il linguaggio Java come base per le spiegazioni e la sperimentazione.

Per permettere agli studenti e alle studentesse di sperimentare le nozioni apprese durante le ore di teoria in aula l'insegnamento include una sostanziale parte di laboratorio. I temi introdotti durante il laboratorio corredano e integrano le conoscenze derivanti dalla parte teorica (knowledge and understanding) e permettono di familiarizzare con le metodologie e tecnologie introdotte, anche investigando soluzioni alternative (applying knowledge and understanding). Inoltre durante le ore di laboratorio è previsto lo sviluppo di un’applicazione distribuita realistica con interfaccia grafica. La preparazione e la discussione del progetto di laboratorio sono inoltre volte a stimolare le capacità di organizzare il lavoro in piccoli gruppi, e poi di illustrare verbalmente le soluzioni adottate (communication skills).

3. Risultati dell'apprendimento attesi:

Gli studenti e le studentesse acquisiranno la conoscenza dei concetti che stanno alla base della programmazione di interfacce utente grafiche e di applicazioni distribuite e concorrenti in linguaggi ad alto livello. Essi/esse dovranno essere in grado di sviluppare applicazioni distribuite in

ambiente Java, utilizzando RMI e socket Java per la realizzazione di applicazioni distribuite, la programmazione a thread per la gestione del parallelismo e SWING e JavaFX per la realizzazione delle interfacce utente grafiche.

4. Modalità di verifica dell'apprendimento:

L'esame è composto da un TEST SCRITTO e da una VERIFICA DI LABORATORIO. Le due prove possono essere sostenute in qualsiasi ordine (cioè, non è necessario aver dato lo scritto per fare la prova di laboratorio, né il vice versa).

- TEST SCRITTO: Prova scritta che include esercizi e domande teoriche sul programma dell'insegnamento. Viene valutata da un minimo di 0 ad un massimo di 30 e si considerano sufficienti i voti >=18. Come da regolamento di Ateneo, ogni studente/studentessa può sostenere un numero massimo di tre prove scritte durante l'Anno Accademico (cioè, consegnare il proprio elaborato tre volte). Il voto ottenuto durante un test scritto decade se si partecipa ad un altro test scritto e si consegna l'elaborato.
- VERIFICA DI LABORATORIO: prevede la discussione del progetto di laboratorio svolto durante il corso. La discussione deve essere effettuata preferibilmente in unica soluzione, con tutti i membri del gruppo di laboratorio presenti, e si può tenere sia durante gli appelli appositamente dedicati che su appuntamento (previa email al/alla docente). Il voto di laboratorio è un numero compreso tra 0 e 30, si considerano sufficienti i voti >=18.
- CALCOLO DEL VOTO FINALE DI ESAME: sia X il voto del test scritto; sia Y il voto di laboratorio. Il voto Fin finale dell'esame si ottiene come segue: Fin = (X+Y)/2

Note: i voti acquisiti durante la prova di laboratorio, o durante il test scritto, rimangono validi fino al termine della terza sessione d’esame (quella che precede l'inizio del nuovo insegnamento). Quando si superano entrambe le prove, è necessario registrare il voto finale entro i limiti imposti dal Regolamento di Ateneo.

5. Modalità d'insegnamento:

L’insegnamento è diviso in una parte di teoria e una di laboratorio. Per la parte di teoria sono previste 40 ore di lezione frontali che seguono il programma riportato più avanti, integrate da casi di studio e da esercitazioni volte ad illustrare l’applicazione pratica dei concetti appena studiati. La parte di laboratorio si tiene in altre 20 ore e ha per oggetto l’insegnamento delle tecnologie per lo sviluppo di applicazioni distribuite e concorrenti in Java (RMI, thread), con interfacce grafiche a finestre (SWING e JavaFX/FXML). Le lezioni si svolgono in maniera interattiva e sono corredate da vari esercizi miranti a fornire esempi pratici.

Le sperimentazioni che vengono effettuate durante le ore di laboratorio, strutturate come sequenze di esercizi specifici, sono fondamentali per aiutare gli studenti e le studentesse a comprendere e assimilare i contenuti teorici spiegati a lezione in quanto permettono di mettere in pratica i concetti e le metodologie illustrate su esempi concreti. Inoltre lo sviluppo del progetto di laboratorio, organizzato in gruppi di max 4 studenti/studentesse, permette di consolidare le conoscenze teoriche in un caso realistico di media complessità e di esercitarsi nel lavoro collaborativo.

Si consiglia caldamente la frequenza costante alle lezioni di teoria e di laboratorio. E’ inoltre fondamentale iscriversi al corso online su I-Learn, all’interno del quale viene messo a disposizione materiale didattico di supporto.

6. Attività di supporto:

Il materiale didattico di supporto (lucidi delle lezioni, link a documentazione, esempi di testo di esami ed altro) è disponibile presso il supporto on-line ai corsi: qui

Si noti che i lucidi non sostituiscono il libro di testo, né il materiale integrativo ad esso affiancato.

7. Programma:

Programmazione ad eventi in Java: programmare interfacce grafiche.
- Sorgenti di eventi, gestori di eventi, event-driven programming.
- Organizzazione e uso delle interfacce grafiche di Java.

- L'architettura Model-View-Controller (MVC).

Programmazione Multithread:
- Esecuzione concorrente di istruzioni.
- I thread in Java: ciclo di vita dei thread.
- Creazione e sincronizzazione di thread.
- Estensione del modello della memoria in presenza di thread.
- Problemi di sincronizzazione e loro risoluzione mediante il linguaggio Java.

Programmazione in rete in Java:
- L'architettura client-server.
- Uso di socket.
- Polimorfismo e trasferimento di oggetti mediante Java.
- Invocazione remota di metodi (RMI).
- Il modello di esecuzione distribuita di oggetti.

8. Testi consigliati e bibliografia:

Libro di Testo: Silvia Crafa. Oggetti, concorrenza, distribuzione. Società editrice Esculapio, 2014.

Altri testi (approfondimenti):

- K. Arnold, J. Gosling, D. Holmes. Il linguaggio Java, Manuale Ufficiale. Pearson Education Italia, 2006.
- Programmazione Java - tecniche avanzate, di Deitel e Deitel. Ed. Pearson - Prentice Hall

Insegnamento **MFN0635 - Reti di Elaboratori**

Insegnamento (inglese):	
CFU:	**12**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1** **2**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Michele GARETTO (Titolare)** **Matteo SERENO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

- Competenza nella programmazione sequenziale C e/o Java (per recepire gli elementi di programmazione concorrente e distribuita presentati nel corso);
- Competenza nelle architetture degli elaboratori e dei sistemi operativi (per apprezzare gli aspetti sistemistici dei sistemi distribuiti oggetto di studio).

Eventuali corsi propedeutici

Programmazione I e II, Architetture, Sistemi Operativi.

2. Obiettivi formativi:

Il corso studia gli elementi fondamentali delle tecnologie di trasmissione del livello data link, dei protocolli di accesso a mezzi condivisi e dei protocolli di trasmissione wireless, la suite di protocolli TCP/IP, e i principi che guidano la strutturazione e la progettazione di applicazioni distribuite.

3. Risultati dell'apprendimento attesi:

- Comprensione dei fenomeni e dei protocolli che sono alla base della trasmissione delle informazioni via cavo e via etere;
- Conoscenza approfondita dei protocolli che compongono la suite TCP/IP e delle loro relazioni;
- Principi che guidano la strutturazione e la progettazione di applicazioni distribuite.

4. Modalità di verifica dell'apprendimento:

La verifica si articolerà in due prove scritte che potranno richiedere complessivamente tra le tre e le quattro ore. La verifica sarà orientata ad accertare la capacità maturata dallo studente nella comprensione dei fenomeni e dei protocolli di livello data link, di accesso a mezzo condiviso, di trasmissione wireless, nella conoscenza approfondita dei protocolli che compongono la suite TCP/IP e delle loro relazioni, e nella capacità di progettare e realizzare semplici applicazioni distribuite.

5. Modalità d'insegnamento:

Le lezioni e le esercitazioni sono svolte con l'ausilio di slide che verranno messe a disposizione sul sito di e-learning del corso di laurea.

6. Attività di supporto:

Il materiale didattico di supporto è disponibile sul siti e-learning del Corso di Laurea.

7. Programma:

- Introduzione ad Internet ed alle reti di calcolatori
- Il livello di collegamento e le reti locali: collegamenti, reti di accesso e reti locali
- Reti Wireless e Mobili
- Il livello di Rete
- Il livello di Trasporto
- Il livello Applicazioni
- Come creare un’applicazione di rete: I socket
- Gestione della rete

8. Testi consigliati e bibliografia:

Libri di testo:

- Reti di Calcolatori e Internet - Un approccio top-down, J. Kurose, K. Ross, Pearson Education, Sesta Edizione.
- Introduzione alla programmazione client-server, D. Maggiorini, Pearson, Addison Wesley

Libri consigliati:

- Internetworking con TCP/IP. Vol. 1: Principi, protocolli e architetture, D. Comer, Pearson Prentice Hall

Insegnamento **MFN1362 - Reti I**

Insegnamento (inglese):
CFU: **6**
Settore:
Periodo didattico: **1**
Tipologia di Attività Formativa: **B - caratterizzante**

Docenti: **Marco BOTTA (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Lo studente deve conoscere prima di seguire il corso i fondamenti della programmazione e dei sistemi operativi.

Eventuali corsi propedeutici

Sistemi Operativi, Programmazione I e II

2. Obiettivi formativi:

Il corso si propone di fornire agli studenti nozioni base sulle reti di calcolatori e una comprensione approfondita della suite di protocolli TCP/IP. Inoltre, attraverso l'uso di software ed esempi pratici, il corso fornisce agli studenti una comprensione concreta dei meccanismi di comunicazione tra dispositivi di rete e calcolatori.

3. Risultati dell'apprendimento attesi:

Il corso si propone di preparare gli studenti allo sviluppo di un progetto di tipo generico che coinvolga una tecnologia di rete relativa alla famiglia di protocolli TCP/IP. Al tempo stesso, gli studenti che superano con successo l'esame potranno inserirsi anche in gruppi di lavoro tecnici che debbano affrontare problematiche di sviluppo di applicazioni distribuite di progettazione, implementazione e/o adeguamento di protocolli di rete.

4. Modalità di verifica dell'apprendimento:

La verifica viene fatta sulle conoscenze acquisite durante il corso. La durata della prova scritta è di 2 ore e prevede domande relative agli argomenti trattati nel Corso. L'esame può avere un seguito in forma orale, che è facoltativo per lo studente.

5. Modalità d'insegnamento:

Le lezioni sono di tipo frontale in aula con esempi dal PC del docente collegato in rete e al proiettore durante la lezione

6. Attività di supporto:

Diverso materiale reperibile tramite: Reti e Sistemi Distribuiti

7. Programma:

Fondamenti su reti di calcolatori.

Parte introduttiva alle reti (i livello di comunicazione fisico, il livello link, reti wireless e mobilità) Il livello rete: instradamento e reti IP Il livello transport: controllo della congestione e del flusso end-to-end Il livello applicativo: suite di applicazioni TCP/IP (Web, Posta, DNS, File transfer, sistemi P2P).

8. Testi consigliati e bibliografia:

J. Kurose - K. Ross. Computer Networking: A Top-Down approach featuring the Internet. 6/e Pearson - Addison Wesley

Edizione italiana: J. Kurose - K. Ross. Reti di calcolatori e internet - Un approccio top-down 6a Edizione Pearson - Addison Wesley

Testo supplementari: D. Comer. Internetworking with TCP/IP Volume Unico, Prentice Hall

Insegnamento **INF0002 - Servizi Web**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**D - libera**
Docenti:	**Liliana ARDISSONO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Si richiede una buona conoscenza delle basi di dati (fornita dall'insegnamento Basi di Dati), dei sistemi operativi (fornita dall'insegnamento Sistemi Operativi), dell'analisi e della progettazione object-oriented (fornita da Programmazione II e di Algoritmi e strutture dati) e dei fondamenti della programmazione distribuita (fornita dall'insegnamento Programmazione III). Gli studenti e le studentesse devono inoltre avere la capacità di scrivere, compilare e verificare la correttezza di programmi in Java.

Eventuali corsi propedeutici

Basi di Dati, Sistemi Operativi, Interazione Uomo-Macchina, Programmazione II, Algoritmi e Strutture Dati, Programmazione III.

2. Obiettivi formativi:

L’implementazione delle applicazioni web richiede di guardare “dietro all’interfaccia utente” per andare a fondo su aspetti architetturali e tecnologici che possono influenzare non solo le prestazioni, la scalabilità e la robustezza, ma anche le tipologie di servizio che possono essere effettivamente offerte.

L'insegnamento si pone come obiettivo di fornire la conoscenza di base necessaria per la progettazione e lo sviluppo di applicazioni Web interattive, accessibili da terminali desktop e mobili (grazie all’uso di linguaggi di interfaccia utente cross-platform), e caratterizzate da una logica applicativa mediamente complessa. In ultimo ci si propone di formare programmatori/programmatrici capaci di sviluppare applicazioni web di qualità e basate su architetture standard, largamente utilizzate nel mondo aziendale.

Le tecnologie presentate sono note come Server-side Programming e riguardano la progettazione e lo sviluppo di applicazioni basate su architetture modulari che possono accedere a sorgenti dati eterogenee (come basi dati relazionali, file, etc.) allo scopo di fornire agli/alle utenti servizi complessi. Più precisamente, l'insegnamento tratterà dal punto di vista sia teorico che pratico: (i) lo sviluppo di pagine web statiche e dinamiche responsive; (ii) la programmazione lato server. Inoltre, l'insegnamento tratterà la rappresentazione e interpretazione dei dati in XML, data la sua importanza nella gestione dell'interoperabilità tra applicazioni.

Per permettere agli studenti e alle studentesse di sperimentare le nozioni apprese durante le ore di teoria in aula l'insegnamento include una sostanziale parte di laboratorio. I temi introdotti durante il laboratorio corredano e integrano le conoscenze derivanti dalla parte teorica (knowledge and understanding) e permettono agli studenti e alle studentesse di familiarizzare con le metodologie e tecnologie introdotte, anche investigando soluzioni alternative (applying knowledge and understanding). Inoltre durante le ore di laboratorio è previsto lo sviluppo di un’applicazione realistica con interfaccia web. La preparazione e la discussione del progetto di laboratorio sono inoltre volte a stimolare le capacità di organizzare il lavoro in piccoli gruppi, e poi di illustrare verbalmente le soluzioni adottate (communication skills).

3. Risultati dell'apprendimento attesi:

Gli studenti e le studentesse acquisiranno la conoscenza delle architetture di riferimento per lo sviluppo di applicazioni web modulari. Inoltre acquisiranno la conoscenza dei modelli più usati per gestire applicazioni interattive di media complessità, e delle tecnologie attualmente utilizzate per l'implementazione delle applicazioni e della loro interfaccia utente. Gli studenti e le studentesse acquisiranno la capacità di sviluppare pagine dinamiche PHP, e applicazioni Web a 3 livelli in ambiente Java, utilizzando HTML5-JavaScript, CSS, JSP e Java Servlet, e il framework MVC Angular JS.

4. Modalità di verifica dell'apprendimento:

L'esame è composto da un TEST SCRITTO e da una VERIFICA DI LABORATORIO. Le due prove possono essere sostenute in qualsiasi ordine.

TEST SCRITTO

Prova scritta che include esercizi e domande teoriche sul programma dell'insegnamento. Viene valutata da un minimo di 0 ad un massimo di 30 e si considerano sufficienti i voti >=18. Come da regolamento di Ateneo, ogni studente/studentessa può sostenere un numero massimo di 3 prove scritte durante l'Anno Accademico (cioè, consegnare il proprio elaborato 3 volte). Il voto ottenuto durante un test scritto decade se si partecipa ad un altro test scritto e si consegna l'elaborato.

VERIFICA DI LABORATORIO

Prevede la discussione del progetto di laboratorio svolto durante il corso. La discussione deve essere effettuata preferibilmente in unica soluzione, con tutti i membri del gruppo di laboratorio presenti. Il voto di laboratorio è un numero compreso tra 0 e 30 e si considerano sufficienti i voti >=18.

CALCOLO DEL VOTO FINALE DI ESAME

Sia X il voto del test scritto; sia Y il voto di laboratorio. Il voto Fin finale dell'esame si ottiene come segue: Fin = (X+Y)/2

NB: i voti acquisiti durante la prova di laboratorio, o durante il test scritto, rimangono validi fino al termine della terza sessione d'esame (quella che precede l'inizio del nuovo corso). Quando si superano entrambe le prove, è necessario registrare il voto finale entro i limiti imposti dal Regolamento di Ateneo.

5. Modalità d'insegnamento:

L'insegnamento è diviso in una parte di teoria e una di laboratorio.

Per la parte di teoria sono previste 30 ore di lezione frontali, che seguono il programma dell'insegnamento, integrate da casi di studio e da esercitazioni volte ad illustrare l'applicazione pratica dei concetti appena studiati.

La parte di laboratorio si tiene in altre 30 ore e ha per oggetto l'insegnamento e la sperimentazione di tecnologie per lo sviluppo di pagine web statiche e dinamiche, e di applicazioni server-side MVC con accesso a basi di dati relazionali. Le esercitazioni propongono vari esercizi mirati a fornire esempi pratici, e sono svolte a calcolatore, utilizzando ambienti di sviluppo professionali, come Eclipse e/o NetBeans.

Le sperimentazioni che vengono effettuate durante le ore di laboratorio, strutturate come sequenze di esercizi specifici, sono fondamentali per aiutare gli studenti e le studentesse a comprendere e assimilare i contenuti teorici spiegati a lezione in quanto permettono di mettere in pratica i concetti e le metodologie illustrate su esempi concreti. Inoltre lo sviluppo del progetto di laboratorio, che si effettua in gruppo (max 3 persone per gruppo), permette di consolidare le conoscenze teoriche e il lavoro in team in un caso realistico di media complessità.

Si consiglia caldamente la frequenza costante alle lezioni di teoria e di laboratorio. E' inoltre fondamentale iscriversi al corso online su I-Learn, all'interno del quale viene messo a disposizione materiale didattico di supporto.

6. Attività di supporto:

Il materiale didattico di supporto (lucidi delle lezioni, link a documentazione, esempi di testo di esami ed altro) è disponibile presso il supporto on-line ai corsi: qui

Si noti che i lucidi non sostituiscono il libro di testo, né il materiale integrativo ad esso affiancato.

7. Programma:

Architetture delle applicazioni Web: Web browser e Web server.

Il Pattern Model View Controller per le applicazioni Web - Progettazione e sviluppo di applicazioni Web a 3 livelli basate su MVC:

- Il primo livello (client dell'applicazione): HTML5, CSS, scripting lato client (JavaScript e AJAX), Java Server Pages (JSP). Raccolta dati con HTML form.
- Il secondo livello (logica applicativa): Servlet Java.
- Il terzo livello (livello dei dati): accesso a database relazionali con Java Database Connectivity (JDBC), rappresentazione e gestione di informazioni con XML (XML Schema, XPath, XML Parsers).

Framework MVC Angular JS per le applicazioni MVC.

Introduzione al linguaggio PHP per lo sviluppo di pagine web dinamiche.

8. Testi consigliati e bibliografia:

Libro di testo: Programmazione Web lato Server, di V. Della Mea, L. Di Gaspero, I. Scagnetto, Apogeo, 2007

Altri testi (per gli interessati):

- Programmazione Java - tecniche avanzate, di Deitel e Deitel. Ed. Pearson - Prentice Hall
- Principi di Web design, di Joel Sklar. Ed. Apogeo

Insegnamento **MFN0636 - Sicurezza**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**B - caratterizzante** **D - libera**
Docenti:	**Francesco BERGADANO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Si presuppone la conoscenza dei sistemi operativi e delle reti di calcolatori basate sui protocolli della suite TCP/IP. La conoscenza di specifiche piattaforme di sviluppo e programmazione, quali Java e C++, considerata di aiuto alla comprensione degli argomenti svolti nel corso.

Eventuali corsi propedeutici

Sistemi Operativi, Reti di Calcolatori

2. Obiettivi formativi:

Il corso si propone di fornire agli studenti gli strumenti crittografici e tecnici utilizzati per garantire la sicurezza di reti e calcolatori. Inoltre, attraverso l'uso di esempi pratici, il corso fornisce agli studenti una comprensione concreta dei maggiori rischi di sicurezza e delle soluzioni disponibili

3. Risultati dell'apprendimento attesi:

Il corso si propone di preparare gli studenti a lavorare in azienda per la gestione dei sistemi informatici, cooperando con i livelli organizzativi per garantire la sicurezza del sistema informativo nel suo insieme.

4. Modalità di verifica dell'apprendimento:

La verifica viene fatta sulle conoscenze acquisite durante il corso. La durata della prova scritta e' di un'ora e prevede domande aperte e domande a risposta multipla, relative agli argomenti trattati nel corso

5. Modalità d'insegnamento:

Le lezioni sono di tipo frontale in aula con esempi dal PC del docente collegato in rete e al proiettore durante la lezione.

6. Attività di supporto:

slide online per alcune parti del corso, per il resto si fa riferimento al libro di testo.

7. Programma:

Strumenti crittografici: cifrari simmetrici e asimmetrici, funzioni di hash, firma elettronica
Sicurezza della rete privata: analisi dei rischi di sicurezza informativa, controllo di accesso, protezione da virus, sistemi firewall, reti private virtuali

8. Testi consigliati e bibliografia:

Libro di testo: William Stallings: Cryptography and Network Security Prentice Hall, Seconda

Edizione, 1998

Altri Testi per consultazione: Bruce Schneier: Applied Cryptography, John Wiley and sons, 1994.
David Curry: Unix System Security, Addison-Wesley, 1992.

Insegnamento	**MFN0618 - Sistemi Informativi**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**D - libera**
Docenti:	**Roberto MICALIZIO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Si richiede allo studente di padroneggiare i concetti relativi alle basi di dati relazionali e alla modellazione concettuale (Entity-Relationship), alla programmazione e allo studio degli algoritmi.

Eventuali corsi propedeutici

Basi Dati e Sperimentazioni.

2. Obiettivi formativi:

L'insegnamento di Sistemi Informativi ha lo scopo di dare una panoramica delle maggiori e più diffuse applicazioni delle basi di dati nel mondo del lavoro e dell'impresa in cui buona parte dei processi aziendali sono ormai automatizzati. A tal fine, il corso introdurrà nozioni basilari Business Process Management (BPM) e consentirà di sperimentare la modellazione di processi di business mediante lo standard Business Process Modeling Notation (BPMN).
Inoltre, l'insegnamento introduce lo studente ai sistemi di pianificazione delle risorse aziendali (Enterprise Resource Planning - ERP), ai sistemi integrati di gestione e profilazione del cliente (CRM), ai sistemi di supporto alla decisione (DSS) e alle piattaforme di Business Intelligence. In particolare, l'insegnamento si focalizzerà sulle Data Warehouse e le metodologie di analisi on-line dei dati (OLAP).

3. Risultati dell'apprendimento attesi:

Conoscenza delle principali ambiti di applicazione dei sistemi informativi. Capacità di progettazione di processi di business supportati dai sistemi informativi. Progettazione ed interrogazione delle Data Warehouse.

4. Modalità di verifica dell'apprendimento:

L'esame è composto da due parti. La prima parte consiste in un progetto di laboratorio in cui lo studente progetta un sistema informativo, un processo aziendale da esso supportato, e una data warehouse correlata. La seconda parte è una prova scritta in cui saranno verificate con domande aperte ed esercizi le competenze acquisite dallo studente.

5. Modalità d'insegnamento:

Le lezioni si svolgono in aula con l'ausilio di slides e di materiale distribuito via Moodle. Le esercitazioni si volgeranno in laboratorio tramite una suite di strumenti software open source (come Bonita BPM, PostgreSQL.

6. Attività di supporto:

Si veda il sito Moodle del corso a cura del docente.

7. Programma:

1. Introduzione ai sistemi informativi aziendali (Modello organizzativo, modello funzionale, modello informatico)

2. Sistemi ERP (Le suite ERP; Paradigma ERP; Piattaforme software; Offerta ERP; Trasformazione dell’impresa)

3. Integrazione con il cliente: i sistemi CRM (Ruolo dei sistemi CRM nelle aziende; Schema architetturale; Il paradigma; Esempi; Suite di package software; Evoluzione dei CRM)

4. Piattaforme di Business Intelligence e DSS (12 ore) (Livello delle fonti; Data warehouse; ETL; Progettazione del sistema di warehousing; Livello di elaborazione: reporting e DSS; Motori di analisi e data mining; Suite software per i sistemi direzionali; Sistemi CRM analitici)

5. Gestione dell'azienda orientata ai processi (Studio e sperimentazione di diversi linguaggi di modellazione quali: UML,e BPMN)

8. Testi consigliati e bibliografia:

- G. Bracchi, C. Francalanci, G. Motta, Sistemi informativi d'impresa, McGraw-Hill, Milano, 2010, ISBN: 978-88-386-6328-4
- M. Golfarelli, S. Rizzi, Data Warehouse - Teoria e pratica della progettazione 2/ed, McGraw-Hill, 2006, ISBN: 9788838662911

Libro di consultazione per la parte di Data Warehouse:

- Ralph Kimball, Margy Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, John Wiley & Sons, 2nd Edition, 2002

Libro di consultazione per la parte di Business Process Management:

- Antonio Di Leva, La Gestione dell’Azienda Basata sui Processi e i Sistemi Informativi Aziendali, Celid, 1st Edition, 2014, ISBN 978-88-6789-017-0

Insegnamento **MFN0607 - Sistemi Intelligenti**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**B - caratterizzante** **D - libera**
Docenti:	**Cristina BAROGLIO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Poiché Sistemi Intelligenti è il primo corso che tratta argomenti di Intelligenza Artificiale, le competenze attese in ingresso riguardano competenze nel settore informatico. In particolare: - conoscenza di algoritmi su alberi e grafi con relative nozioni di complessità - esperienza di programmazione con particolare riferimento a programmazione ad oggetti (organizzazione in classi e sottoclassi, ereditarietà) - nozioni di logica (calcolo proposizionale e calcolo dei predicati del primo ordine) - nozioni di modelli semantici dei dati nelle basi dati.

Eventuali corsi propedeutici

Gli studenti che sono iscritti al corso di laurea di Informatica di Torino acquisiscono le competenze in ingresso sopra elencate seguendo gli insegnamenti di: - “Algoritmi e strutture dati”, - “Programmazione I e II”, - “Basi di dati”, - “Matematica Discreta e Logica” (e sostenendo i relativi esami)

2. Obiettivi formativi:

Il corso si propone di fornire una introduzione generale alle problematiche nel settore dell'Intelligenza Artificiale, con particolare attenzione a come sia possibile costruire un sistema dotato di capacità autonome di risoluzione di problemi, di ragionamento e di apprendimento quando abbia a disposizione una rappresentazione simbolica del mondo.

Il corso si articola in tre parti principali: - Risoluzione automatica di problemi - Rappresentazione della conoscenza e ragionamento - Nozione di agente intelligente che agisce, ragiona ed apprende

Data la natura introduttiva del corso e la durata del corso, molte problematiche avanzate di Intelligenza Artificiale trovano collocazione nei corsi offerti per l'indirizzo "Sistemi per il Trattamento dell'Informazione" della laurea magistrale in Informatica

3. Risultati dell'apprendimento attesi:

ITALIANO

Dal punto di vista della acquisizione di contenuti metodologici, gli studenti conosceranno le principali strategie di ricerca per la risoluzione automatica di problemi, i fondamenti della rappresentazione della conoscenza mediante formalismi logici e approcci strutturati (ontologie) e relativi meccanismi inferenziali.

Avranno inoltre acquisito le nozioni di base su architettura di un agente intelligente e su apprendimento automatico di conoscenza a partire da esempi.

Gli studenti acquisteranno una prima forma di consapevolezza sulla necessità di fare dei trade-off (in termini di qualità della soluzione e costo computazionale). tra strategie e metodologie alternative che risolvono la stessa classe di problemi.

L’esame orale richiede inoltre allo studente di acquisire una buona (discreta) capacità espositiva unita all’uso corretto della terminologia del settore (e più in generale dell’informatica).

ENGLISH

For what concerns the acquisition of methodological skills, students will learn the main search strategies for problem solving, basic knowledge about knowledge representation through logic formalisms and structered approaches (ontologies) and also the corresponding inference mechanisms.

They will also learn basic notions on agent architectures and on supervised machine learning.

Students will gain awareness of the need of a trade-off (in terms of solution quality and computational cost) between alternative strategies and methodologies for solving a same class of problems.

The oral examination requires also that students acquire good presentation skills and that they acquire competence in the correct use of the specialized vocabulary of the sector as well as of computer science in general.

4. Modalità di verifica dell'apprendimento:

ITALIANO

L'esame consiste in una interrogazione individuale sugli argomenti trattati nel corso. Tipicamente una delle domande richiede la capacità di applicare una metodologia/strategia per risolvere un semplice problema simile a quelli illustrati a lezione. Pertanto nell’esame orale, oltre a verificare la conoscenza dei concetti e delle metodologie, si intende verificare la capacità di applicazione dei concetti e metodologie a problemi specifici. Essendo l’esame un orale, è possibile inoltre verificare la chiarezza di esposizione e l’uso corretto della terminologia.

ENGLISH

The exam consists of an individual interview on the course topics. Typically one of the questions concerns the application of one of the methodologies/approaches studied during the course on a problem similar to one of those studied. Thus, the exam does not only verify the conceptual knowledge students have built but also the ability of practically using that knowledge. Moreover, the nature of the examination allows verifying presentation clarity and the correct use of terminology.

5. Modalità d'insegnamento:

ITALIANO

Le lezioni si svolgono in aula. Oltre alle lezione tipiche frontali in cui il docente illustra gli aspetti metodologici, nell’insegnamento numerose ore sono dedicate ad esempi ed esercitazioni (svolte dal docente). Vengono anche illustrati alcuni esempi significativi di strumenti software e di applicazioni di Intelligenza artificiale a problemi complessi del mondo reale. Vengono anche messe in evidenza possibili problematiche etiche.

ENGLISH

Lectures take place in a classroom. They concern both explanation of methodological aspects as well as exercises, solved collaboratively under the lead of the teacher, and discussions of many examples. Some relevant tools are shown and their basic functioning is explained; some more complex tools that are/were used for tackling real world problems are also explained. Ethical concerns are also underlined.

6. Attività di supporto:

ITALIANO

Verra' reso disponibile sulla apposita pagina del corso all'interno del servizio I LEARN. Tale

materiale prevede ulteriori esempi di utilizzo dei meccanismi di ragionamento e di strategie di ricerca oltre a quelli contenuti sul libro di testo.

ENGLISH

Additional material is made available through the platform I LEARN. This includes further examples of application of reasoning and search strategies.

7. Programma:

Come già detto l'insegnamemnto è una introduzione ai concetti basilari di Intelligenza articiaicile e si articola in tre parti strettamente connesse.

Parte 1) RISOLUZIONE AUTOMATICA DI PROBLEMI In questa parte si affronta la problematica di come definire il concetto di problema e di soluzione, di distinguere tra soluzione e soluzione ottima. Sono studiati tre approcci alla risoluzione di problemi: ricerca nello spazio degli stati, ricerca in spazi con avversario (giochi ad informazione completa), risoluzione di problemi mediante soddisfacimento di vincoli. Per ciascun approccio si discutono le principali strategie di ricerca: ampiezza, profondità, iterative deepening (per le ricerche cieche nello spazio degli stati), A* e Recursive Best First Strategy (per le ricerche euristiche), Min-Max e Alfa-beta (per i giochi con avversario), backtracking, forward propagation e arc consistency per meccanismi basati su soddisfacimento di vincoli. Particolare attenzione viene data alla garanzie offerte dalle diverse strategie in termini di qualità della soluzione e di complessità computazionale.

Parte 2) RAPPRESENTAZIONE DELLA CONOSCENZA E RAGIONAMENTO Il problema della rappresentazione della conoscenza e dei relativi meccanismi inferenziali viene affrontato studiando due principali famiglie di approcci alla rappresentazione della conoscenza: formalismi logici e rappresentazioni strutturate. Per quanto riguarda i formalismi logici si vede come sia il calcolo proposizionale che il calcolo dei predicati del primo ordine possano essere utilizzati per rappresentare conoscenza sul mondo e si vede come i meccanismi inferenziali (modus ponens, resolution, etc.) possano essere adoperati per fornire servizi utili (es. risposta a domande, verifica consistenza, ecc.). Si analizza anche come una rappresentazione a regole permetta meccanismi di ragionamento più efficienti (forward e backward chaining). Notevole attenzione viene data alla rappresentazione della conoscenza strutturata introducendo tassonomie, classi, individui, ereditarietà singola e multipla, inferenze specializzate. Queste nozioni vengono analizzate ed esemplificate mediante uso del linguaggio ontologico OWL2 (proposto e supportato da W3C).

Parte 3) AGENTI E APPRENDIMENTO AUTOMATICO In questa parte conclusiva si introduce la nozione di agente intelligente che opera in un ambiente e si fa vedere come l’agente possa avere sia comportamenti reattivi che deliberativi a seconda del compito assegnato. Si illustra come agente debba avere capacità di risoluzione automatica di problemi e di ragionamento sullo stato del mondo e sul suo stato. Si descrive brevemente come l’apprendimento automatico sia una delle caratteristiche essenziali per ottenere un agente intelligente. Vengono introdotte solo nozioni elementari con particolare riguardo all'apprendimento da esempi (in particolare apprendimento di alberi di decisione). Viene infine fatta una introduzione alle reti neuronali come strumento per passare dal livello sub simbolico a quello simbolico.

8. Testi consigliati e bibliografia:

S. J. Russell e P. Norvig, Artificial Intelligence: a modern approach , Third edition, 2010, Pearson Nell’ambito del corso verranno affrontate le problematiche descritte nei capitoli 1,2,3,5,6,7,8,9, 18 del testo. Si ricorda che esiste anche la traduzione italiana della prima parte della terza edizione (volume 1) e la traduzione integrale della seconda edizione (volume 1 e 2), entrambe edite da Pearson/Prentice Hall.

OWL 2 Web Ontology Language Primer (Second Edition) http://www.w3.org/TR/owl2-primer/

Insegnamento **MFN0601 - Sistemi Operativi**

Insegnamento (inglese):	
CFU:	**12**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Cristina BAROGLIO (Titolare)** **Enrico BINI (Titolare)** **Massimiliano DE PIERRO (Titolare)** **Daniele GUNETTI (Titolare)** **Daniele Paolo RADICIONI (Titolare)** **Claudio SCHIFANELLA (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Agli studenti è richiesta una conoscenza di base dell'architettura di un computer (secondo quanto studiato nel corso di Architetture degli Elaboratori I) e dei concetti di base di programmazione (secondo quanto studiato nel corso di Programmazione I). Gli studenti dovranno inoltre essere in grado di padroneggiare i sistemi di enumerazione binario (base due) ed esadecimale (base sedici).

Eventuali corsi propedeutici

Costituiscono prerequisiti i contenuti dei corsi di:

* Architettura degli Elaboratori I * Programmazione I

2. Obiettivi formativi:

Il sistema operativo costituisce l'interfaccia fondamentale tra l'utilizzatore di un computer e il computer stesso. Parte essenziale del curriculum di base di un laureato in informatica è la conoscenza di come il sistema operativo sia in grado di amministrare le varie componenti hardware di cui è composto un computer. Queste modalità di amministrazione devono essere il più possibile trasparenti al generico utilizzatore del computer, ma devono essere conosciute a fondo da ogni specialista del settore. L'insegnamento fornisce dunque una conoscenza di base dell'architettura interna e del funzionamento dei moderni sistemi operativi, e di come, ai fini di garantire un ragionevole compromesso tra efficienza, sicurezza e facilità d'uso, vengono amministrate le risorse fondamentali della macchina su cui il sistema operativo è installato: il processore, la memoria principale e la memoria secondaria.

Per la parte di laboratorio gli obiettivi formativi sono l'apprendimento del linguaggio C, utilizzato per la programmazione nell'ambiente del sistema operativo Unix. La parte di laboratorio mira a fornire allo studente una conoscenza (teorica e pratica) di base sui comandi della shell, sulla gestione dei processi, sugli strumenti di inter-process communication e sulla gestione dei segnali forniti dal sistema, oltre che alcuni rudimenti di programmazione bash.

I temi introdotti durante il laboratorio corredano e integrano le conoscenze derivanti dalla parte teorica (knowledge and understanding), al tempo stesso presentando esempi di problemi realistici di comunicazione e sincronizzazione su cui gli studenti sono sollecitati a cimentarsi, anche investigando soluzioni alternative (applying knowledge and understanding). La preparazione e la discussione del progetto sono inoltre volte a stimolare le capacità di organizzare il lavoro in piccoli gruppi (2-3 studenti), e poi di illustrare verbalmente le soluzioni adottate (communication skills).

- Corso A: TUTTE LE INFORMAZIONI SUL CORSO E IL MATERIALE DIDATTICO SI TROVERANNO

ALL'URL:

Sistemi Operativi

- Corso B: TUTTE LE INFORMAZIONI SUL CORSO E IL MATERIALE DIDATTICO SI TROVERANNO IN MOODLE

3. Risultati dell'apprendimento attesi:

Lo studente acquisirà la conoscenza dell'architettura e del funzionamento dei moderni sistemi operativi, e dovrà essere in grado di ragionare sulle prestazioni fornite dal sistema e su eventuali inefficienze. Avrà inoltre appreso i fondamenti della programmazione C e concorrente e la capacità di sviluppare programmi concorrenti in grado di interagire fra loro senza causare anomalie o blocchi del sistema. Infine, avrà una conoscenza di base di come le risorse della macchina (in particolar modo il tempo di CPU, lo spazio di memoria primaria e lo spazio di memoria secondaria) possano essere sfruttate al meglio.

Lo studente dovrà essere in grado di sviluppare programmi scritti nel linguaggio C, e programmi di sistema e concorrenti codificati con i comandi propri dell'ambiente del sistema operativo Unix; dovrà inoltre essere in grado di ragionare su alcuni problemi di comunicazione fra processi e sincronizzazione utilizzando gli strumenti introdotti durante il laboratorio.

4. Modalità di verifica dell'apprendimento:

Per superare l'esame di sistemi operativi lo studente deve ottenere un punteggio sufficiente (almeno 18) nelle prove in cui si articola l'esame, che dovranno essere sostenute nel seguente ordine:

- Prova di laboratorio: consiste nella discussione individuale del progetto di laboratorio ed è finalizzata a valutare le competenze acquisite nei moduli di laboratorio. Più precisamente saranno valutati sia gli strumenti insegnati nel modulo Unix, sia le competenze inerenti il linguaggio C.

- Prova scritta: è finalizzata a valutare le competenze acquisite sugli argomenti insegnati nel modulo di teoria e C. In generale potrà essere composta da domande aperte, domande chiuse a scelta multipla, esercizi.

- Orale facoltativo. Può essere sostenuto solo avendo preso nello scritto almeno 26/30. Integra il voto ottenuto sostenendo le parti di esame 1) e 2), e permette di prendere un voto massimo di 30/30 e lode.

Il voto finale sarà ottenuto come somma pesata dei voti conseguiti nelle prove descritte.

5. Modalità d'insegnamento:

L'insegnamento è diviso in una parte di teoria e una di laboratorio.

Per la parte di teoria sono previste 60 ore di lezione frontali che seguono il programma riportato più avanti, integrate da casi di studio e da esercitazioni volte ad illustrare l'applicazione pratica dei concetti appena studiati.

La parte di laboratorio è articolata in due moduli da 30 ore ciascuno, il primo avente per oggetto l'insegnamento del linguaggio C, e il secondo rivolto alla programmazione in ambiente UNIX. Le lezioni si svolgono in maniera interattiva e sono corredate da vari esercizi miranti a fornire esempi pratici.

6. Attività di supporto:

Dispense e lucidi usati a lezione, programmi di esempio.

- Per la parte di teoria del corso A il materiale è disponibile

qui

- Per la parte di teoria del corso B il materiale sarà disponibile in Moodle

Sono inoltre previste 12 ore facoltative di esercitazione e in cui saranno proposti esercizi e relative soluzioni sugli argomenti principali del corso e saranno revisionati gli aspetti teorici relativi.

7. Programma:

NOTA: Par la parte di teoria, il programma è basato sul TESTO DI RIFERIMENTO.

PARTE DI TEORIA:

* Introduzione al Corso di Sistemi Operativi

* PARTE I: GENERALITA'

o Introduzione (cap. 1)

o Strutture dei Sistemi Operativi (cap. 2)

* PARTE II: GESTIONE DEI PROCESSI

o Processi (cap. 3)

o Thread (cap. 4)

o Scheduling della CPU (cap. 5)

o Sincronizzazione dei Processi (cap. 6)

o Deadlock (Stallo di Processi) (cap. 7)

* PARTE III: GESTIONE DELLA MEMORIA (PRIMARIA)

o Memoria Centrale (cap. 8)

o Memoria Virtuale (cap. 9)

* PARTE IV: GESTIONE DELLA MEMORIA SECONDARIA

o Interfaccia del File System (cap. 10)

o Realizzazione del File System (cap. 11)

o Memoria Secondaria e Terziaria (Gestione dell'Hard disk)

==========

PARTE DI LABORATORIO:

- Linguaggio C
- Introduzione a Unix (comandi, shell, file system, diritti d'accesso, ridirezione, pipe)
- Make e makefile
- System call per la creazione e la sincronizzazione di processi
- System call per L'InterProcess Communication e per la gestione di segnali
- Esercitazioni pratiche, in particolare: esercitazioni finalizzate ad imparare il linguaggio C, ad utilizzare Unix e a sviluppare programmi concorrenti

8. Testi consigliati e bibliografia:

Teoria (TESTO DI RIFERIMENTO):

Silberschatz-Galvin-Gagne. Sistemi Operativi - ottava o nona edizione – Agli studenti sono anche resi disponibili fin dall’inizio del corso i lucidi usati dal docente nelle lezioni di teoria.

Laboratorio: • A. Kelley e I. Pohl: "C didattica e programmazione" (presente in Biblioteca) • R. Stevens, S. Rago: "Advanced Programming in the UNIX Environment", 2a ed., Addison Wesley • M.J. Bach: "The Design of the UNIX Operating System", Prentice-Hall International Editions (L3167), per la parte relativa all'implementazione del kernel Unix. • S.R. Bourne: "The Unix System" (L1856, L1803, L2140) • K. Christian: "The Unix Operating System", John Wiley & Sons (L3147)

Insegnamento **INF0004 - Storia dell'Informatica**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1** **2**
Tipologia di Attività Formativa:	**D - libera**
Docenti:	**Felice CARDONE (Titolare)** **Daniele GUNETTI (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Conoscenza di base dell'architettura di un computer e di un sistema operativo. Dimestichezza di base con i principali paradigmi di programmazione.

Eventuali corsi propedeutici

Architetture degli elaboratori I. Programmazione I e II. Sistemi Operativi.

2. Obiettivi formativi:

Dare una visione di insieme e una prospettiva storica dei momenti salienti dell'evoluzione delle idee teoriche e pratiche dell'informatica.

3. Risultati dell'apprendimento attesi:

Acquisire una visione di insieme e una prospettiva storica dei momenti salienti dell'evoluzione delle idee teoriche e pratiche dell'informatica.

4. Modalità di verifica dell'apprendimento:

Esame scritto

5. Modalità d'insegnamento:

Orale

6. Attività di supporto:

7. Programma:

Il corso fornisce una introduzione alla storia dell’informatica, dedicando particolare attenzione ad alcune tappe che hanno fornito un contributo essenziale alla definizione della forma attuale dell’informatica (per esempio, lo sviluppo di Internet con il suo retroterra culturale e tecnologico). L’interesse centrale del corso è per le idee e la loro evoluzione, più che per le singole innovazioni tecnologiche: non si tratta di una rassegna di modelli di macchina calcolatrice, ma di una introduzione ai modi di pensiero, alle metafore ed alle visioni che hanno caratterizzato la scienza dell’informazione e del calcolo attraverso la storia.

La prima parte del corso sarà dedicata ad una storia dell’evoluzione delle architetture, dei sistemi operativi e dei linguaggi di programmazione, insieme ad alcuni cenni di storia dell’informatica commerciale e di Internet.

La seconda parte comprende, accanto ad un’introduzione istituzionale alla storia del calcolo basata su uno dei libri di testo elencati, anche una serie di lezioni monografiche dedicate a temi e lavori classici in varie aree dell’informatica.

L’esame del corso prevede un compito finale il cui scopo sarà quello di verificare la comprensione della portata tecnologica e culturale degli eventi salienti della storia dell’informatica.

AVVISO per gli studenti del DAMS che hanno in piano di studi l'insegnamento di Fondamenti di Informatica II, che quest'anno mutua da Storia dell'informatica. Si intende che la parte mutuata è la seconda, che sarà tenuta nel secondo semestre dal Prof. Cardone.

8. Testi consigliati e bibliografia:

M. Campbell-Kelly and W. Aspray. Computer: A History of the Information Machine. Basic Books, 1996.

P.E. Ceruzzi. A History of Modern Computing (2nd edition ed.). MIT Press, 2003.

M. Davis. Il calcolatore universale. Da Leibniz a Turing. Adelphi, 2012.

M. Priestley. A Science of Operations: Machines, Logic and the Invention of Programming. Springer, 2011.

B. Randell (Ed.), The Origins of Digital Computers: Selected Papers (3rd ed.). Springer-Verlag, 1982.

Insegnamento **MFN0606 - Sviluppo delle Applicazioni Software**

Insegnamento (inglese):	
CFU:	**9**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**2**
Tipologia di Attività Formativa:	**B - caratterizzante**
Docenti:	**Matteo BALDONI (Titolare)** **Sara CAPECCHI (Titolare)** **Claudia PICARDI (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Programmazione orientata agli oggetti (Programmazione I e II)

Rudimenti di progettazione e interrogazione di database (Basi di Dati)

Eventuali corsi propedeutici

Programmazione I e II. Basi di Dati

2. Obiettivi formativi:

Il corso si propone di introdurre gli studenti ai concetti di base dell'ingegneria del software e allo sviluppo di applicazioni software, utilizzando la metodologia Agile Unified Process (UP), che sfrutta il linguaggio di modellazione UML. Lo studente dovrà saper sviluppare un'applicazione significativa individuando con chiarezza la logica applicativa, l'interazione con le basi di dati e le interfacce richieste dai requisiti. Inoltre dovrà imparare a pianificare il lavoro secondo i canoni dello sviluppo dei progetti: lavoro di gruppo, definizione degli obiettivi e delle fasi di sviluppo.

Il corso ha una forte caratterizzazione sperimentale. Per questa ragione è suddiviso in un modulo di 4 CFU di lezioni frontali e una modulo di laboratorio di 5 CFU.

3. Risultati dell'apprendimento attesi:

Lo studente acquisirà:

- conoscenze di base sull'ingegneria del software;
- conoscenza della metodologia UP e dell'uso di UML in tale ambito;
- capacità di applicare UP alla progettazione di un applicativo;
- conoscenza di design pattern di progettazione e architetturali;
- capacità di individuare l'utilità e l'applicabilità dei design pattern al progetto di un software.

4. Modalità di verifica dell'apprendimento:

L'esame consiste in una parte di teoria e una parte di progetto. La parte di progetto viene svolta in gruppo in parte durante il laboratorio e in parte nel proprio tempo di studio individuale. La valutazione del progetto avviene tramite una discussione di gruppo. La valutazione di tale parte è espressa in trentesimi.

La parte di teoria consiste di una prova scritta, opzionalmente seguito da una prova orale, che dunque resta facoltativa. La valutazione per questa parte è espressa in trentesimi.

La votazione finale è data dalla media fra i due voti ottenuti nelle due parti.

Le due parti si possono sostenere insieme (nello stesso appello) oppure separatamente, in qualsiasi ordine.

5. Modalità d'insegnamento:

L’insegnamento è diviso in una parte di teoria e una di laboratorio.

Per la parte di teoria sono previste 40 ore di lezione frontale che seguono il programma, integrate da casi di studio e da esercitazioni volte ad illustrare l’applicazione pratica dei concetti studiati.

La parte di laboratorio è composta da 50 ore e consiste nello sviluppo guidato di un progetto di medie dimensioni, realizzato in gruppi utilizzando la metodologia UP.

6. Attività di supporto:

Vengono effettuate consegne periodiche degli artefatti UP per il progetto di laboratorio, che man mano vengono corretti e commentati durante le ore di laboratorio.

7. Programma:

1) Elementi di ingegneria del software: modelli Waterfall, Spirale, V-shaped, Componend-based Development, metodologie Agili tra cui: SCRUM e extreme programming.

2) Introduzione all'UML: use case diagram, class diagram, object diagram, sequence diagram, communication diagram, state chart, activity diagram.

3) Una metodologia Agile: Unified Process (UP). Tale metodologia verrà descritta in dettaglio a lezione durante le ore di teoria e applicata a un caso di studio nel laboratorio. I passi previsti, in breve, sono:

- Pianificazione delle fasi di sviluppo: esse sono suddivise in ideazione, elaborazione e costruzione, possiamo pensare ad esempio a una iterazione di ideazione, due iterazioni di elaborazione e una iterazione di costruzione. Di tutte le iterazioni si prevede una durata e i documenti prodotti (tabella delle attività).
- Prima iterazione (di ideazione). A partire da una descrizione informale del progetto da sviluppare, si comincia l’analisi dei requisiti con: scelta degli attori e descrizione dei loro obiettivi, individuazione dei casi d’uso (documento prodotto: use case diagram). Si fa poi una suddivisione fra i casi d’uso a seconda della loro priorità (ad esempio: alta, media, bassa), secondo dei criteri da fissare a priori. Si produce il class diagram di dominio. Si dettagliano i casi d’uso ad alta priorità, si fanno i sequence diagram di sistema per individuare le operazioni che corrisponderanno a eventi legati all’interfaccia utente e si scrivono i contratti di quelle operazioni che si ritengono più complesse. Si producono anche il documento di visione, il documento delle specifiche supplementari, la prima versione del glossario.
- Seconda e terza iterazione (di elaborazione). Durante queste iterazioni si eseguono sia attività di design e implementazione che di analisi. Per quanto riguarda la progettazione, si sceglie la classe Controller (la classe nel sistema che “parla” con l’interfaccia utente). Si dettagliano i corpi delle operazioni dei casi d’uso ad alta priorità tramite sequence o communication diagram. Si produce un secondo class diagram, in cui si dettagliano campi e metodi e si applicano i design pattern per organizzare il software. Si cominciano a progettare la base di dati e l’interfaccia utente. Si cominciano a sviluppare gli unit test e l’implementazione dei casi d’uso ad alta priorità.

Per quanto riguarda l’analisi, si rivedono i casi d’uso sviluppati nel passo precedente. Si dettagliano i casi d’uso a priorità media e il class diagram di dominio. Si fanno i sequence diagram di sistema e i contratti per tali casi d’uso. Si aggiorna il glossario.

- Quarta iterazione (di costruzione). Si progettano e si implementano i casi d'uso a media priorità. Si rivedono eventualmente quelli ad alta priorità. Si procede con gli unit test (insieme eventualmente ad altri test). Si fa il deployment.

4) Testing: unit testing, acceptance test, white e black box testing, controllo delle versioni.

8. Testi consigliati e bibliografia:

Craig Larman. Applicare UML e i pattern. Terza edizione (seconda edizione italiana), Pearson (Capitoli 1-21; Capitoli 23-26; cenni ai Capitoli 28, 29, 30, 31, 33, 34; Capitolo 36; cenni al Capitolo 38).

Roger S. Pressman, Bruce R. Maxim. Software Engineering: A Practitioner's Approach, 8/e. McGraw-Hill.

Ian Sommerville. Software Engineering, 10/e. Pearson.

Altri testi consigliati:

Fowler. UML Distilled. Terza edizione italiana. Pearson-Addison Wesley.

Gamma, Helm, Johnson, Vlissides. Design pattern. Prima edizione italiana. Pearson Education Italia-Addison Wesley.

NOTE Software consigliato per UML: UMLET.

Il materiale del corso consiste in slide, testi e soluzioni di esercizi, esempi, riferimenti bibliografici e articoli di approfondimento per la parte di teoria; in piu', per la parte di laboratorio, sono forniti artefatti completi e semi-lavorati appartenenti al progetto proposto. Tutto il materiale viene pubblicato sulla piattaforma I-learn (http://informatica.i-learn.unito.it/).

Insegnamento **MFN0634 - Tecnologie Web**

Insegnamento (inglese):	
CFU:	**6**
Settore:	**INF/01 - INFORMATICA**
Periodo didattico:	**1**
Tipologia di Attività Formativa:	**B - caratterizzante** **D - libera**
Docenti:	**Giancarlo Francesco RUFFO (Titolare)**

1. Prerequisiti e Propedeuticità:

Competenze attese in ingresso

Lo studente deve mostrare di possedere una buona familiarità con i principi della progammazione (imperativa, ad oggetti e basata su eventi). Inoltre, deve conoscere le basi operative per gestire una base di dati basata su SQL e per configurare/installare pacchetti software nel proprio sistema operativo.

Eventuali corsi propedeutici

- MFN0582 - Programmazione I
- MFN0585 - Programmazione II
- MFN0601 - Sistemi Operativi
- MFN0602 - Basi di Dati

2. Obiettivi formativi:

Gli obiettivi di questo corso sono i seguenti:

- Imparare a produrre siti Web dinamici, animati, interattivi e collegati ad un database in back end;
- Imparare diversi linguaggi e tecnologie per lo sviluppo Web client-side, quali HTML5, CSS, JavaScript, JQuery
- Imparare principi della programmazione server side tramite principalmente PHP e MySQL, sfruttando strumenti opensource come i comuni browser web e il server web Apache.

3. Risultati dell'apprendimento attesi:

HTML5, PhP, JavaScript, CSS, Web services, mySQL, version control

4. Modalità di verifica dell'apprendimento:

Esercizi di laboratorio (almeno 4) (10%): saranno corretti e valutati i vari esercizi di laboratorio consegnati.

Relazione (35%): deve contenere la descrizione del progetto che si intende realizzare e deve essere consegnata almeno un mese prima della data di appello scelta (2000-3000 parole). Conterrà una descrizione funzionale, il wireframe del sito ed altre specifiche che saranno definite dal docente durante il corso.

Progetto finale (55%): progetto individuale che implementa quanto dichiarato dallo studente nella

relazione consegnata precedentemente. Il sito sarà sottoposto alle opportune fasi di test e valutato di conseguenza.

Gli studenti non frequentanti che non potranno consegnare gli esercizi di laboratorio, potranno sostenere un esame orale sostitutivo basato sul progetto consegnato.

5. Modalità d'insegnamento:

Le lezioni teoriche saranno svolte in modalità tradizionale/frontale, con l'ausilio di diapositive che saranno proiettate in aula. La parte applicativa del corso sarà svolta in laboratorio informatico e gli studenti saranno incentivati a svolgere gli esercizi in aula e consegnarli per la correzione. Le diapositive saranno messe a disposizione degli studenti come materiale integrativo.

6. Attività di supporto:

Il materiale didattico di supporto sarà reso tramite un'apposita pagina web accessibile tramite il sistema di supporto alla didattica Moodle gestito dal nostro Corso di Laurea.

7. Programma:

- Progettazione base ed implementazione di siti Web
- Presentazione delle diverse strategie di navigazione e di organizzazione dei siti
- Tecnologie client-side, tra cui HTML5, CSS, Javascript, JSON e JQuery
- Tecnologie server side, facendo particolare attenzione alle implementazioni in PhP
- Gestione dei dati in back end
- Tecnologie emergenti (MVC, bootstrap, angular, versioning con github, etc.)

8. Testi consigliati e bibliografia:

J. Miller, V. Kirst, Marty Stepp. [Web Programming Step by Step http://www.webstepbook.com/index.sh... 2nd edition (2012)