Informatica A – Seconda Prova in Itinere

**Cognome Matricola**

**Nome**

Istruzioni

1. Non separate questi fogli. Scrivete la soluzione **solo** **sui fogli distribuiti**, utilizzando il retro delle pagine in caso di necessità. **Cancellate le parti di brutta** (o ripudiate) con un tratto di **penna**.
2. Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.
3. **È possibile scrivere a matita** (e non ricalcare al momento della consegna!).
4. È **vietato** utilizzare **calcolatrici** o **telefoni**. Chi tenti di farlo vedrà **annullata** la sua prova.
5. È ammessa la consultazione di **libri** e **appunti**, purché con pacata discrezione e senza disturbare.
6. Qualsiasi **tentativo** di comunicare con altri studenti comporta **l’espulsione** dall’aula.
7. È possibile **ritirarsi senza penalità**.
8. Non è possibile lasciare l’aula conservando il tema della prova in corso.
9. Tempo a disposizione:
10. **1h:00m per recupero prima prova**
11. **2h:15m per seconda prova**

**RECUPERO PRIMA PROVA**

**Esercizio 1.1 ( 6 punti )**

**Esercizio 1.2 ( 8 punti )**

**SECONDA PROVA**

**Esercizio 2.1 ( 4 punti )**

**Esercizio 2.2 ( 4 punti )**

**Esercizio 2.3 ( 6 punti )**

**Esercizio 2.4 ( 4 punti )**

**Voto finale:**

Esercizio 1.1 ( 6 punti ) – RECUPERO PRIMA PROVA

I dati di un’università sono organizzati nel seguente modo: gli studenti, i corsi e gli esami sono disposti in array. Le strutture dati utilizzate sono le seguenti:

#define N 1000

#define NS 20000

#define NC 2000

#define NE 400000

typedef struct { int giorno, mese, anno; } Data;

typedef struct { char matricola[7], nome[N], cognome[N];

Data dataNascita;

int valido; } Studente;

typedef Studente Studenti[NS];

typedef struct { char codice[11], titolo[N];

int numCrediti;

int valido; } Corso;

typedef Corso Corsi[NC];

typedef struct { char codiceCorso[11], matricolaStudente[N];

int voto;

Data data;

int valido; } Esame;

typedef Esame Esami[NE];

Il campo “valido” serve a dire se la casella dell’array contiene contenuto valido (nel caso l’attributo ha valore 1) o è da considerarsi vuota (nel caso l’attributo ha valore 0).

Si codifichi in C la seguente funzione:

void f(Esami es, Corsi cor, Studenti stu, Studenti stuBravi)

che riceve i tre array *es*, *stu* e *cor* contenenti i dati dell’università e l’array *stuBravi* inizialmente tutto con caselle non valide e copia (senza lasciare buchi) in *stuBravi* i dati degli studenti che hanno la media superiore a 27.

Esercizio 1.2 ( 8 punti ) – RECUPERO PRIMA PROVA

Si scriva una funzione f che riceve in input due array di caratteri

int f(char str[], char numero[])

L’array numero contiene una stringa fatta da solo caratteri 1 o 0 e rappresenta un numero binario.

La funzione deve convertire il numero binario rappresentato nell’array numero in decimale e verificare se il numero di vocali contenute nel vettore str è pari al valore del numero convertito. Se è così restituisce 1, altrimenti restituisce 0.

Esercizio 2.1 ( 4 punti ) SECONDA PROVA

Si dice cosa stampa il seguente codice:

#define dim 5

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

typedef struct item {

char dato;

struct item \* next;

} Item;

typedef Item \* lista;

lista mistero(lista L) {

lista temp = L;

while (temp->next != NULL)

temp = temp->next;

temp->next = (lista) malloc(sizeof(Item));

temp->next->next = NULL;

temp->next->dato = L->dato + 8;

return L->next;

}

int main(void) {

lista L, temp;

int i;

L = (lista) malloc(sizeof(Item));

L->next = NULL;

L->dato = 'e';

for (i=1; i<dim; i++) {

temp = (lista) malloc(sizeof(Item));

temp->next = L;

temp->dato = 'm' - i\*4;

L = temp;

}

L = mistero(L);

while (L != NULL) {

printf("%c ", L->dato);

L = L->next;

}

getch();

return (0);

}

Esercizio 2.2 ( 4 punti ) SECONDA PROVA

Il seguente schema descrive i dati di una nota università milanese

Studente ( Matricola, Nome, Cognome, DataNascita, AnnoDiCorso )

Esame ( Matr, CodCorso, Data, Voto )

Corso ( Codice, NomeCorso, CodiceDocente, Anno, CFU )

Docente ( CodiceDocente, Nome, Cognome, DataNascita, LuogoNascita )

Estrarre in SQL i docenti di un corso che avevano preso 18 quando, da studenti, ebbero a sostenere l’esame del corso di cui ora sono titolari (come un certo A.C. nel corso di Informatica A). Si consideri sufficientemente certa l’identificazione di una persona in base a nome, cognome e data di nascita.

SELECT S.\*

FROM Studente S JOIN Esame ON Matricola=Matr JOIN Corso C ON C.Codice=CodCorso

JOIN Docente D ON C.CodiceDocente=D.CodiceDocente

WHERE S.Cognome=D.Cognome AND S.Nome=D.Nome AND S.DataNascita=D.DataNascita

AND Voto=18

Il seguente schema descrive i dati di una nota università milanese

Studente ( Matricola, Nome, Cognome, DataNascita, AnnoDiCorso )

Esame ( Matr, CodCorso, Data, Voto )

Corso ( Codice, NomeCorso, CodiceDocente, Anno, CFU )

Docente ( CodiceDocente, Nome, Cognome, DataNascita, LuogoNascita )

Estrarre in SQL gli studenti che non hanno mai dato un esame il giorno del loro compleanno.  
Si ricorda che le funzioni day e month estraggono giorno e mese da una data. Il confronto sul compleanno è quindi

day(data)=day(dataNascita) and month(data)=month(dataNascita)

SELECT \*

FROM Studente

WHERE Matricola NOT IN SELECT Matricola

FROM Studente S JOIN Esame E ON Matr=Matricola

WHERE day(data)=day(dataNascita) and month(data)=month(dataNascita)

Esercizio 2.3 ( 6 punti ) SECONDA PROVA

I dati di un’università sono organizzati nel seguente modo: gli studenti, i corsi e gli esami sono disposti in liste. Le strutture dati utilizzate sono le seguenti:

#define N 1000

typedef struct { int giorno, mese, anno; } Data;

typedef struct S { char matricola[7], nome[N], cognome[N];

Data dataNascita;

S \* next; } Studente;

typedef Studente \* ListaStudenti;

typedef struct C { char codice[11], titolo[N];

int numCrediti;

C \* next; } Corso;

typedef Corso \* ListaCorsi;

typedef struct E { char codiceCorso[11], matricolaStudente[N];

int voto;

Data data;

E \* next; } Esame;

typedef Esame \* ListaEsami;

Si codifichi in C la seguente funzione:

ListaStudenti f(ListaEsami es, ListaCorsi cor, ListaStudenti stu)

che riceve le tre liste *es*, *stu* e *cor* contenenti i dati dell’università e restituisce (senza intaccare i parametri di ingresso) la listadegli studenti che hanno la media superiore a 27.

int creditiCorso( char codiceCorso[], ListaCorsi LC) {

while(LC!=NULL)

if(strcmp(codiceCorso,LC->codice)==0)

return LC->crediti;

else LC=LC->next;

return 0;

}

float media( char Matr[], ListaEsami LE, ListaCorsi LC) {

float media=0; int crediti=0,cr;

while(LE!=NULL) {

if(strcmp(Matr,LE-> matricolaStudente)==0) {

cr= creditiCorso( LE->codiceCorso, LC);

media+=LE->voto \* cr;

crediti+=cr;

}

LE=LE->next;

}

if(crediti>0)

return media/crediti;

else return 0;

}

ListaStudenti f(ListaEsami es, ListaCorsi cor, ListaStudenti stu) {

ListaStudenti result=NULL;

while(stu!=NULL) {

if(media( stu->Matr[], es, cor)>27)

result=insInFondo(result,stu);

}

return result;

}

ListaStudenti **InsInFondo**( ListaStudenti lista, ListaStudenti elem ) {

ListaStudenti punt, cur = lista;

punt = (ListaStudenti) malloc( sizeof(Studente) );

punt–>next = NULL;

strcpy(punt–>matricola,elem->matricola); /\* Crea il nuovo nodo \*/

strcpy(punt–>nome,elem->nome); /\* Crea il nuovo nodo \*/

strcpy(punt–>cognome,elem->cognome); /\* Crea il nuovo nodo \*/

punt-> dataNascita=elem-> dataNascita;

if ( lista==NULL )

return punt; /\* => punt è la nuova lista \*/

else {

while(cur–>next!= NULL ) /\* Trova l’ultimo nodo \*/

cur = cur–>next;

cur –>next = punt; /\* Aggancio all’ultimo nodo \*/

}

return lista;

}

Si codifichi in C la funzione:

ListaEsami eliminaStudentiFalsi(ListaEsami es, ListaStudenti stu)

che elimina da *es* i dati di studenti non presenti nella lista *stu*.

Esercizio **2.4** ( 4 punti ) – SECONDA PROVA

Si consideri la seguente definizione di un albero binario:

typedef struct EL { int dato;

struct EL \* left, \* right; } node;

typedef node \* tree;

Implementare una funzione che, ricevuto in ingresso un albero binario, restituisce il numero di valori distinti contenuti nell’albero. È consigliato implementare funzioni di supporto.