Informatica A

**Cognome**

**Nome**

**Matricola o Codice studente**

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: 2 ore e 40 minuti

**Esercizio 1 ( 2 punti )**

**Esercizio 2 ( 4 punti )**

**Esercizio 3 ( 4 punti )**

**Esercizio 4 ( 9 punti )**

**Esercizio 5 ( 7 punti )**

**Esercizio 6 ( 6 punti )**

**Voto finale:**

Esercizio 1 ( 2 punti )

Si costruisca la tabella di verità della seguente espressione booleana.

( C and ( not B ) ) and ( ( not A ) and B)

Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i numeri A = -101 e B = 28, li si converta, se ne calcoli la differenza (A-B) in complemento a due e si indichi se si genera riporto sulla colonna dei bit più significativi e se si verifica overflow.

Esercizio 2 ( 4 punti )

Si consideri il seguente schema di base di dati relativa ad autoveicoli:

MARCA(NOME, NAZIONE)

TIPOVEICOLO(NOMEMARCA, MODELLO, PROPULSORE, CONTATORE)

ACQUIRENTE(CODFISC, TARGA, NOMEMARCA, MODELLO, PROPULSORE, CITTADINANZA)

Scrivere una query SQL che estrae per ogni ACQUIRENTE il numero di veicoli acquistati.

MARCA(NOME, NAZIONE)

TIPOVEICOLO(NOMEMARCA, MODELLO, PROPULSORE, CONTATORE)

ACQUIRENTE(CODFISC, TARGA, NOMEMARCA, MODELLO, PROPULSORE, CITTADINANZA)

Scrivere una query SQL che estrae la NAZIONE che produce più MARCHE di veicoli.

Esercizio 3 ( 4 punti )

Si dice cosa stampa il seguente codice

#include <stdio.h>

int f(int x, char \* str1, char \* str2){

int i, j=0, k=0, cont=0;

char temp;

while(str1[j]!='\0')

j++;

while(str2[k]!='\0')

k++;

while(x<j && x<k){

for(i=x;i<j;i++){

temp=str1[i];

str1[i]=str2[k-i-1];

str2[k-i-1]=temp;

cont++;

}

x++;

}

return cont;

}

int main(){

char c,aggettivo[]="btaeo",nome[]="studvnre";

printf("%s %s %d\n", aggettivo, nome, 3\*f(1,aggettivo,nome));

return 0;

}

Esercizio 4 ( 9 punti )

Si consideri un tavolo da biliardo rappresentato da una matrice

int M[R][C];

in cui ogni elemento rappresenta una possibile coordinata all’interno del tavolo. Si supponga che inizialmente tutte le celle contengano il numero di birilli presenti nella coordinata stessa.

Si assuma ora che una palla all’interno del tavolo possa muoversi solo lungo 4 direzioni 45°, 135°, 225°, 315°, che quando colpisce una sponda rimbalzi seguendo le leggi della fisica (e.g. se colpisce una sponda verticale in direzione 45° poi prosegue in direzione 135°). Si ignori il caso in cui la palla colpisce uno degli angoli del biliardo e che dopo aver toccato la quarta sponda si arresti.

Si scriva una funzione tiroMigliore che prende in ingresso la matrice M e due coordinate intere (che supponiamo essere ammissibili) che indicano la posizione della palla. La funzione deve calcolare qual è la “direzione migliore”, cioè quella che corrisponde al tiro che colpisce più birilli, restituire il numero totale di birilli colpiti dalla palla lanciata lungo la direzione migliore e modificare la matrice M in modo da contenere -1 in ogni cella attraversata dalla palla lanciata lungo la direzione migliore.

Si supponga di avere a disposizione la funzione

void direzioni(int angolo, int \* sp\_r, int \* sp\_c);

che dato un angolo, definisce due valori di spostamento lungo le righe e le colonne.

Esercizio 5 ( 7 punti )

Si consideri la seguente definizione di lista:

typedef struct EL {

int dato;

struct EL \* next;

} nodo;

typedef nodo \* lista;

Scrivere una funzione (e tutte le funzioni ausiliarie per facilitare la risoluzione) che prende in ingresso due liste e restituisce una nuova lista contenente i valori che compaiono in entrambe le liste lo stesso numero di volte. La nuova lista non deve contenere valori duplicati (anche se questi compaiono più volte nelle due liste di partenza).

Esercizio **6** ( 6 punti )

Si consideri la seguente definizione di un albero binario

typedef struct nodeS { int v;

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

typedef node \* tree;

Definiamo come grado di una foglia la somma dei valori associati ai nodi sul percorso dalla radice al nodo foglia. Si scriva una funzione che prende in ingresso un albero binario e restituisce 1 se esistono almeno due foglie con uguale grado.

Si consiglia l’uso di funzioni e strutture dati ausiliarie.