Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria

Informatica A - a.a 2018/2019 - 28 Gennaio 2019

Cognome: _____ **Matricola:** _____

Nome: _____ **Firma:** _____

Istruzioni

- 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** con un tratto di **penna**.
- Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.
- **NON è possibile scrivere a matita.**
- È **vietato** utilizzare **calcolatrici, telefoni o pc**. Chi tenti di farlo vedrà **annullata** la sua prova.
- **Non è ammessa la consultazione di libri e appunti.**
- Qualsiasi **tentativo** di comunicare con altri studenti comporta **l'espulsione** dall'aula.
- È possibile ritirarsi senza penalità.
- Non è possibile lasciare l'aula conservando il tema della prova in corso.
- Tempo a disposizione: **2h30m**

Valore indicativo degli esercizi, voti parziali e voto finale:

Esercizio 1	**3 punti**	_____
Esercizio 2	**3 punti**	_____
Esercizio 3	**4 punti**	_____
Esercizio 4	**6 punti**	_____
Esercizio 5	**12 punti**	_____

Totale(28) _____

NOME e COGNOME: _____

Esercizio 1 - Algebra di Boole, Aritmetica Binaria, Codifica delle Informazioni (3 punti)

(a) Si costruisca la tabella di verità della seguente espressione booleana in tre variabili, badando alla precedenza tra gli operatori logici. Eventualmente si aggiungano parentesi. Non si accetteranno soluzioni senza il procedimento. (1 punto)

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

Risposta:

A	B	C	OUT
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

OUT = ABC + !B

(b) Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i numeri A = -64dec e B = 33, li si converta, se ne calcolino la somma (A+B) e 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. Non si accetteranno soluzioni senza il procedimento. (1 punto)

Risposta:
-64d = 1000000b - 6bit + 1 di segno = 7bit
33d = 0100001b - 6 bit + 1 segno
cp2 = 1011111b - cp2 7bit

A + B = -64 + 33 = -31 = 1100001b - No riporto perduto né overflow
A - B = -64 -33 = -97 = [1]0011111b - Riporto perduto con overflow

7 bit va da -64 a 63

NOME e COGNOME: _____

(c) Si converta il numero 37.59375 in virgola fissa e in virgola mobile con codifica IEEE 754, sapendo che 1/2 = 0.5, 1/4 = 0.25, 1/8 = 0.125, 1/16 = 0.0625, 1/32 = 0.03125, 1/64 = 0.015625, e 1/128 = 0.0078125. Non si accetteranno soluzioni senza il procedimento. (1 punto)

Risposta:
6bit + segno
37d = 0100101
0.5 + 0.0625 + 0.03125 = 0.10011
37.59375 = 0100101.10011 ¡- Virgola fissa

segno = 0
mantissa = 1.0010110011 non normalizzata
exp = 5 + 127 = 132d = 10000100

IEEE754
segno (1bit) = 0
esponente (8bit) = 10000100
mantissa (23bit) = 100 1011 0011 0000 0000 0000

NOME e COGNOME: _____

Esercizio 2 - Domanda di teoria (3 punti)

(a) Spiegare che cosa si intende per iterazione e per ricorsione. Quando è necessario, o conveniente, usarle? In generale è più efficiente l'iterazione o la ricorsione?

(b) Implementare una semplice funzione di esempio sia in versione iterativa che in versione ricorsiva.

Risposta:

NOME e COGNOME: _____

Esercizio 3 - Comprensione del Codice (4 punti)

Scrivere cosa stampa a video il seguente codice:

```
#include <stdio.h>
#include <string.h>

char* doSomething(char *a, char *b);
int main(void) {
    char *a;
    char *b;

    int n = 7;
    a = (char*)malloc(sizeof(char)*n);
    strcpy(a,"casale");
    b = doSomething(a,b);
    printf("%s %c\n",a,*b);
    return 0;
}

char* doSomething(char *a, char *b){
    int n = 0;
    char c;
    b=a;
    for (b,c='1',n=1; b<&a[strlen(a)]; b++,c++)
    {
        if ((*b)=='a')
        {
            n++;
            *b = c;
        }
        else if ((*b)=='a'+4)
        {
            n++;
            *b = c-'1'+'3';
        }
        printf("%s %c\n",a,*b);
    }
    b = (char*)malloc(sizeof(char));
    *b = ('0'+n);
    return b;
}
```

printf:

_____ _____
_____ _____
_____ _____
_____ _____
_____ _____
_____ _____
_____ _____
_____ _____
_____ _____
_____ _____
_____ _____
_____ _____
_____ _____

Risposta:
casale c c2sale 2 c2sale s c2s4le 4 c2s4le l c2s4l8 8 c2s4l8 4

NOME e COGNOME: _____

Esercizio 4 - Matlab (6 punti)

Scrivere il codice Matlab che restituisca i valori richiesti.
Attenersi al numero massimo di righe di codice indicato.

(a) Creare un vettore colonna A e un vettore colonna B contenente rispettivamente 6 e 5 valori casuali differenti tra loro con valori da 1 a 10; (max 2 riga) (1 punto)

Risposta:

```
A = randi([1 10],[6 1]);
B = randi([1 10],[1 5]);

oppure

A = randi([1 10],[6 1]);
B = randi([1 10],[5 1]);
```

(b) A partire dai vettori A e B, creare una matrice C di 6 righe e 5 colonne (1 punto).

Risposta:

```
Se una colonna e una riga
C = A*B;
C = A.*B;

Se entrambe colonne:

C = A*B';
C = A.*B';
```

(c) Cancellare le righe che hanno come primo valore un numero inferiore a 10 (max 1 riga) (1 punto).

Risposta:

```
C(C(:,1)<10,:)=[]
```

(d) Calcolare la somma di tutte le colonne dispari (max 1 riga) (1 punto).

Risposta:

```
sum(sum(C(:,1:2:end)))
```

(e) Scrivere una funzione che presa in ingresso la matrice C, crea una nuova matrice ”cornice”, ponendo cioè a zero tutti gli elementi non appartenenti alla prima e ultima riga/colonna. (max 8 righe) (2 punto).

Risposta:

```
function [out] = funzione(in)
    out = in;
    for i=2:1:size(in,1)-1
        for j=2:1:size(in,2)-1
           out(i,j) = 0;
        end
    end
end
```

NOME e COGNOME:

Esercizio 5 - Programmazione C (12 punti)

Si vuole creare un programma per trasformare una serie di immagini in una GIF animata.
Un'immagine è definita da una struttura che contiene tre vettori bidimensionali 100x100 contenete valori che vanno da 0 a 255. Ogni matrice corrispone al colore rosso, verde e blue.
Una GIF animata è costituita da una lista i cui elementi sono composti da un'immagine (i tre vettori bidimensionali) ed un numero che corrispone ai secondi per cui l'immagine viene visualizzata durante l'animazione.

1. Definire le strutture dati necessari per la realizzazione del programma. (1 punto)

2. Scrivere una funzione che prende in ingresso due immagini e controlli se differiscono di almeni N pixel. N viene definito dall'utente ed è un parametro che viene passato alla funzione. I pixel differiscono quando i valori dei vettori bidimensionali nella medesima posizione sono differenti
(es. immRed1[10][10] != immRed2[10][10]). Se le immagini differiscono la funzione ritorna 1, se non differiscono restituisce 0. (2 punti)

3. Scrivere una funzione che presa in ingresso un'immagine e un tempo t, la aggiunge in coda alla lista contenente tutte le immagini della GIF animata. Prima di aggiugnerle deve controllare che l'immagine differisca di almeno 10 pixel dall'immagine precedente (Si consiglia di usare la funzione del punto 2). (3 punti)

4. Si vuole riprodurre la GIF al contrario. Scrivere la funzione che presa in ingresso la lista della GIF, crei una nuova lista inserendo gli elementi al contrario. (3 punti)

5. Scrivere una funzione che presa in ingresso la GIF ed una variabile tempo T, elimini dalla lista tutte le immagini che hanno durata inferiore a T. Non deve essere creata una nuova lista, ma vanno eliminati gli elementi dalla lista di partenza. (3 punti)

Risposta:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define M 5

typedef struct immagine{
    int red[M][M];
    int green[M][M];
    int blue[M][M];
} immagine;

typedef struct imgGif{
    immagine img;
    int tempo;
    struct imgGif *next;
} imgGif;

typedef imgGif* ptrGif;

//Funzioni
void generaMatrici(int matrix[][M], int i);
void stampaMatrice(int matrix[][M]);
immagine generaImmagine(int i);
void stampaGif(ptrGif gif);
```

```
int confrontaImmagini(immagine img1, immagine img2, int n);
ptrGif aggiungiGif(ptrGif gif, immagine img, int tempo);
ptrGif contrarioIt(ptrGif gif);
ptrGif contrario(ptrGif gif, ptrGif contrario);
ptrGif velocizza(ptrGif gif, T);

int main(void) {
    ptrGif gif = NULL;
    ptrGif gifContrario = NULL;

    immagine img1;
    immagine img2;
    int c;
    int r;
    int val;

    srand((unsigned)time(0));

    do{
        printf("\n\nMENU\n");
        printf("L'opzione 2 genere una matrince 5x5 e non 100x100 per velocizzare il tutto.
              Il comando generare una matrice uguale o una casuale (per cui diversa)\n\n");
        printf("1) Confronta immagini\n2) Inserisci immagini\n3) Lista al contrario\n4)
              Velocizza GIF\n5) Stampa gif\n6) ESCI\n\n>> ");
        fpurge(stdin);
        scanf("%d",&c);
        switch (c) {
            case 1:
                //Per confrontrare immagine diffrenti mettere 0 e 0 o 0 e 1 nella generazione
                //Per confrontare immagine uguali mettere 1 e 1
                img1 = generaImmagine(0);
                img2 = generaImmagine(1);
                if (confrontaImmagini(img1, img2, 5))
                    printf("Le immagini sono differenti\n");
                else
                    printf("Le immagini non sono differenti\n");

                break;
            case 2:
                printf("Uguale 0, diversa 1: ");
                scanf("%d",&r);
                img1 = generaImmagine(r);
                printf("Inserisci tempo: ");
                scanf("%d",&r);
                gif = aggiungiGif(gif, img1, r);
                break;

            case 3:
                stampaGif(gif);
                printf("\n\nAl contrario: \n\n");
                //gifContrario = contrario(gif, gifContrario);
                gifContrario = contrarioIt(gif);
                stampaGif(gifContrario);
                break;

            case 4:
                printf("Inserire valore minimo tempo immagine: ");
                scanf("%d",&val);
                gif = velocizza(gif, val);
```

```
                break;

            case 5:
                stampaGif(gif);
            default:
                break;
        }
    }while(c!=6);
    return 0;
}

void generaMatrici(int matrix[][M], int k)
{
    int i,j;
    for (i=0; i<M; i++)
    {
        for (j=0; j<M; j++)
        {
            if (k!=0)
                matrix[i][j]=rand()%256;
            else
                matrix[i][j]=100;

        }
    }
}

void stampaMatrice(int matrix[][M])
{
    int i,j;
    for (i=0; i<M; i++)
    {
        for (j=0; j<M; j++)
        {
            printf("%d ",matrix[i][j]);
        }
        printf("\n");
    }
}

void stampaGif(ptrGif gif)
{
    if (gif==NULL)
        return;
    stampaMatrice(gif->img.blue);
    //stampaMatrice(gif->img.green);
    //stampaMatrice(gif->img.red);
    printf("%d\n",gif->tempo);
    stampaGif(gif->next);
}

immagine generaImmagine(int i){
    immagine img;
    generaMatrici(img.blue, i);
    generaMatrici(img.red, i);
    generaMatrici(img.green, i);
    return img;
}

int confrontaImmagini(immagine img1, immagine img2, int n)
```

```
{
    int i, j;
    int cont = 0;

    for (i=0; i<M; i++)
    {
        for (j=0; j<M; j++)
        {
            if ((img1.blue[i][j] != img2.blue[i][j]) || (img1.red[i][j] != img2.red[i][j])
                || (img1.green[i][j] != img2.green[i][j]))
                cont++;
// if (img1.blue[i][j] != img2.blue[i][j])
// cont++;
// else if (img1.red[i][j] != img2.red[i][j])
// cont++;
// else if (img1.green[i][j] != img2.green[i][j])
// cont++;
        }
    }
    if (cont>n)
        return 1;
    else
        return 0;
}

ptrGif aggiungiGif(ptrGif gif, immagine img, int tempo)
{
    //int i,j;
    if (gif==NULL)
    {
        gif = (ptrGif)malloc(sizeof(imgGif));
// for (i=0; i<M; i++)
// {
// for (j=0; j<M; j++){
// gif->img.red[i][j] = img.red[i][j];
// gif->img.blue[i][j] = img.blue[i][j];
// gif->img.green[i][j] = img.green[i][j];
// }
// }
        gif->img = img;
        gif->tempo = tempo;
        gif->next = NULL;
        return gif;
    }

    if ((gif->next!=NULL) || ((gif->next == NULL) && (confrontaImmagini(gif->img,img,10))))
       gif->next = aggiungiGif(gif->next, img, tempo);
    return gif;
}

ptrGif contrarioIt(ptrGif gif)
{
    ptrGif contr = NULL;
    ptrGif temp = NULL;

    while (gif!=NULL){
        temp = (ptrGif)malloc(sizeof(imgGif));
        temp->img = gif->img;
        temp->tempo = gif->tempo;
        temp->next = contr;
```

```
        contr = temp;
        gif = gif->next;
    }
    return contr;
}

ptrGif contrario(ptrGif gif, ptrGif contr){
    ptrGif temp;
    int i,j;

    if (gif==NULL)
        return contr;

    temp = (ptrGif)malloc(sizeof(imgGif));
    for (i = 0; i<M; i++)
    {
        for (j=0; j<M; j++)
        {
            temp->img.blue[i][j] = gif->img.blue[i][j];
            temp->img.red[i][j] = gif->img.red[i][j];
            temp->img.green[i][j] = gif->img.green[i][j];
        }
    }
    temp->tempo = gif->tempo;
    temp->next = contr;
    temp = contrario(gif->next, temp);
    return temp;
}

ptrGif velocizza(ptrGif gif, int T){
    ptrGif temp;
    if (gif==NULL)
        return NULL;
    if (gif->tempo<T)
    {
        temp = gif->next;
        free(gif);
        temp = velocizza(temp, T);
        return temp;
    }
    else
    {
        gif->next = velocizza(gif->next, T);
        return gif;
    }
}
```

NOME e COGNOME: _____

NOME e COGNOME: _____