





Politecnico di Milano

## Dipartimento di Elettronica, Informazione e Bioingegneria

**Informatica A - Prof. A. Fuggetta - a.a 2021/2022 - 27 Agosto 2022**

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.**
- Scrivere nome e cognome su tutti i fogli. Non verranno corretti compiti non firmati o con nome illeggibile
- 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: **2.30h**

### Valore indicativo degli esercizi, voti parziali e voto finale:

|                    |                |       |
|--------------------|----------------|-------|
| <b>Esercizio 1</b> | <b>5 punti</b> | _____ |
| <b>Esercizio 2</b> | <b>6 punti</b> | _____ |
| <b>Esercizio 3</b> | <b>7 punti</b> | _____ |
| <b>Esercizio 4</b> | <b>5 punti</b> | _____ |
| <b>Esercizio 5</b> | <b>9 punti</b> | _____ |
| <b>Totale(32)</b>  |                | _____ |

**Esercizio 1 - Algebra di Boole, Aritmetica Binaria, Codifica delle Informazioni (5 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.  
Scrivere l'espressione semplificata. Non si accetteranno soluzioni senza il procedimento. (2 punto)

$A \text{ and } B \text{ or } A \text{ and } (\text{not } B \text{ or } A \text{ and not } (B \text{ and not } C))$

**Risposta:**

Raccolgo subito A  
 $A (B + \text{not } B + A \dots)$   
 $B + \text{NOT } B = 1$   
 L'equazione equivale ad A

- (b) Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i numeri  $A = -32_{\text{dec}}$  e  $B = 21_{\text{dec}}$  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. (2 punto)

**Risposta:**

$A = -32 = 100000b$   
 $B = 21 = 010101b$   
 $-B = -21 = 101011b$

$A+B$   
 $100000$   
 $010101$   


---

 $110101$

NO Overflow e No Riporto perduto

$A-B$

100000

101011

\_\_\_\_\_

001011

Riporto perduto e overflow

- (c) Si converta il numero 12,625 in virgola fissa e in virgola mobile con codifica IEEE 754 con precisione singola. Non si accetteranno soluzioni senza il procedimento. (2 punto)

**Risposta:**  $12 = 01100$

$0.625 = 0.101$

Virgola fissa: 01100.101

Virgola mobile:

$S = 0$

$M = 1001010000000000000000$

$E = 127 + 3 = 130 = 10000010$

**Esercizio 2 - Teoria (6 punti)**

- (a) Data una CPU a 8 bit in complemento a 2, si vuole eseguire la somma tra i numeri ESADECIMALI 10hex e 70hex

Quale delle seguenti affermazioni sono vere

- ☐ viene visualizzato il valore decimale 128 x
- ☐ viene visualizzato il valore esadecimale 80
- ☐ viene visualizzato il valore decimale 100
- ☐ viene visualizzato il valore esadecimale 100
- ☐ viene visualizzato il valore decimale 80

- (b) Data l'operazione precedente, quale delle seguenti affermazioni è corretta:

- ☐ È attivo solo il bit di overflow x
- ☐ È attivo il bit di carry
- ☐ È attivo il bit di segno x
- ☐ Nessuna delle precedenti

- (c) L'intervallo di valori rappresentabili con 6 bit, utilizzando la rappresentazione in complemento a 2 è:

- ☐ da -32dec a 31dec x
- ☐ da -31dec a 32dec
- ☐ da -20hex a 1Fhex x
- ☐ da -16hex a 15hex

- (d) Una variabile di tipo *double*?

- ☐ ha un ingombro di memoria pari a quello di una variabile di tipo float
- ☐ non ammette un indirizzo di memoria
- ☐ ha un valore massimo pari alla metà di quello del long double
- ☐ ha ingombro di memoria pari a 8 byte x
- ☐ è in gradi di memorizzare senza approssimazione il valore 0.6 in binario

- (e) Dato il prototipo di funzione

`float* f(float uno, int *due);`

e le dichiarazioni di variabili

`float *x;`  
`int *y;`

stabilire quale delle seguenti invocazioni di funzione rispetta il prototipo.

- ☐ `x=f(y,*y)`
- ☐ `x=f(*x,y) ;`
- ☐ `*y = f(x, x)`
- ☐ `y = f(*y, y) ;`
- ☐ `y=f(y,y)`

(f) Si consideri il seguente programma.

```
#include <stdio.h>
void func (int x, int *y) {
    x++;
    (*y)++;
}
int main () {
    int a = 1, b = 2;
    func (a, &b);
    printf ("%d", a + b);
    return 0;
}
```

Cosa stampa la funzione printf?

- ☐ 2
- ☐ 3
- ☐ 4 |—
- ☐ 5

(g) **Matlab (7 punti)**

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

1. Creare una matrice quadrata di dimensione 6 contenente numeri casuali da 1 a 100. (1 riga - 1 punto)

**Risposta:**

```
A = randi([1,100],6);
```

2. Dividere per due tutte le celle pari. (1 riga - 1 punto)

**Risposta:**

```
A(2:2:end,2:2:end) = A(2:2:end,2:2:end)/2;
oppure
A(mod(A,2)==0) = A(mod(A,2)==0)/2;
```

3. Cancellare le colonne con media inferiore a 5 (1 riga - 1 punto)

**Risposta:**

```
A(:,mean(A,1)<5) = [];
```

4. Aggiungere una riga all'inizio della matrice contenente in ogni cella il valore della somma totale della matrice (1 riga - 1 punto)

**Risposta:**

```
A = [sum(A,'all')*ones(1,size(A,2));A]
```

5. Creare un array contenente la media delle colonne (1 riga - 1 punto)

**Risposta:**

```
B = mean(A,1)
```

6. Scrivere una funzione Matlab che presa in ingresso la matrice e l'array delle medie calcolato al punto 5, restituisca l'array A contenente il numero di valori di ogni colonna superiore alla media della colonna. (2 punti)

**Risposta:**

```
Function x = funzioneMedia(A,B)\  
  
    x = [];  
    for (i=1:1:size(A,2))  
        For (j=1:1:size(A,1))  
  
            if (A(i,j)>B(i))  
                x=[x S(i,j)]  
            end  
        End  
    End
```



(h) **Programmazione C (5 punti)**

Scrivere un sottoprogramma baricentro che ricevuto in ingresso un array di valori interi e qualsiasi altro parametro ritenuto strettamente necessario restituisce l'indice dell'elemento che fa da baricentro agli elementi dell'array, -1 se non esiste. Definiamo baricentro di un array di h elementi, l'elemento in posizione i tale per cui:

$$v[0] + v[1] + \dots + v[i] = v[i+1] + v[i+2] + \dots + v[h-1]$$

ovvero la somma degli elementi dell'array da 0 a i è uguale alla somma degli elementi dell'array da i+1 fino alla fine.

**Risposta:**

```
#include <stdio.h>

int baricentro(int array[], int l);

int main(int argc, const char * argv[]) {
    int a[] = {1,2,3,5,9,2,0};
    printf("Baricentro = %d\n",baricentro(a,7));
    return 0;
}

int baricentro(int array[], int l){
    int i,j;
    int somma = 0;

    for (i=1; i<l-1; i++){
        somma = 0;
        for (j=0; j<=i; j++)
            somma += array[j];
        for (j=i+1; j<l; j++)
            somma -= array[j];
        if (somma == 0)
            return i;
    }
    return -1;
}
```

(i) **Programmazione C Liste (9 punti)**

Si ipotizzi di dovere realizzare un software per la gestione delle letture realizzate da più sensori di temperatura e umidità all'interno di serre per la floricoltura.

Una lettura è composta da l'orario in cui è stata effettuata (HH:MM:SS), una stringa che indica il nome del sensore ovvero la posizione in cui è stata effettuata la lettura (es. "Serra1", "Tulipani"...), ed i valori stessi delle letture in gradi Celsius e percentuale di umidità.

L'immagazzinamento delle letture deve avvenire mediante una lista, ordinata per orario di lettura. L'inserimento di una nuova misura viene fatto in coda. La lista può contenere al massimo 7200 misurazioni; una volta raggiunto il massimo delle misure, inserendo un nuovo nodo in coda viene automaticamente eliminato il primo nodo in testa.

Si assuma che, inserendo sempre il nuovo nodo in coda, **la lista sia già sempre ordinata.**

Svolgere l'esercizio attenendosi a quanto richiesto. **NON È RICHiesto SCRIVERE IL MAIN.**

1. Si definiscano le strutture dati necessarie allo sviluppo di questo programma. (1 punto)
2. Scrivere la funzione RICORSIVA `int contaNodi(ptrLista testa)` che conta il numero di nodi presenti nella lista. (2 punti)
3. Facendo uso della funzione al punto precedente, scrivere la funzione `ptrLista aggiungiLetture(ptrLista testa, int ora, int minuto, int secondo, char *locale, int valore)` per inserire un nuovo nodo. Se sono presenti già 7200 letture, la prima lettura viene automaticamente cancellata. (3 punti)
4. Scrivere una funzione che riceve due orari ed elimina dalla lista tutte le letture avvenute nella finestra temporale definita dai due orari. Si ipotizzi che il primo orario sia sempre "anteriore" il secondo orario. `ptrLista rimuoviLetture(ptrLista int daOra, int aOra)` (3 punti)
5. (SOLO PER CHI NON HA SUPERATO IL LABORATORIO)  
Si definisca la funzione `elimina_nodo` che presa in ingresso la lista, il nome del sensore ed un numero `n`, elimini dalla lista le prime `n` registrazioni.

**Risposta:**

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

#define MAX 7200

typedef struct nodo {
    char orario[9];
    char nome[20];
    float gradi;
    float umidita;
    struct nodo *next;
} nodo;

typedef nodo *ptrNode;

int contaNodi(ptrNode lista);
ptrNode aggiungiLetture(ptrNode lista, int ora, int min, int sec, char nome[], float valC, float valH);
ptrNode rimuoviLetture(ptrNode lista, char daOra[], char aOra[]);
ptrNode eliminaLista(ptrNode lista, int n, char *nome);
```

```

void stampaLista(ptrNode lista);

int main(int argc, const char * argv[]) {
    ptrNode lista = NULL;
    char nome[20];
    int r;
    do{
        printf("Menu\n");
        printf("1) Inserisci\n");
        printf("2) Rimuovi\n");
        printf("3) Stampa\n");
        printf("4) ESCI\n");
        scanf("%d",&r);
        switch(r){
            case 1:
                lista = aggiungiLettura(lista, 1, 1, 1, "1", 1, 1);
                lista = aggiungiLettura(lista, 1, 1, 2, "2", 1, 1);
                lista = aggiungiLettura(lista, 1, 1, 3, "3", 1, 1);
                lista = aggiungiLettura(lista, 1, 1, 4, "4", 1, 1);
                lista = aggiungiLettura(lista, 1, 1, 5, "5", 1, 1);
                lista = aggiungiLettura(lista, 1, 1, 6, "6", 1, 1);
                stampaLista(lista);
                break;
            case 2:
                lista = rimuoviLettture(lista, "1:1:2", "1:1:4");
                break;
            case 3:
                stampaLista(lista);
                break;

        }
    }while(r<4);
    return 0;
}

int contNodi(ptrNode lista){
    if (lista==NULL)
        return 0;
    return 1+contNodi(lista->next);
}

ptrNode aggiungiLettura(ptrNode lista, int ora, int min, int sec, char nome[], float valC, float valH){
    ptrNode temp = lista;
    if (lista == NULL)
    {
        lista = (ptrNode)malloc(sizeof(nodo));
        sprintf(lista->orario,"%d:%d:%d",ora,min,sec);
        strcpy(lista->nome,nome);
        lista->gradi = valC;
        lista->umidita = valH;
        lista->next = NULL;
        return lista;
    }

    if (contNodi(lista)>=MAX)
    {
        temp = lista;
        lista = lista->next;
        free(temp);
    }
    temp = lista;
}

```

```

    while(temp->next!=NULL){
        temp = temp->next;
    }

    temp->next = (ptrNode)malloc(sizeof(nodo));
    temp = temp->next;
    sprintf(temp->orario, "%d:%d:%d",ora,min,sec);
    strcpy(temp->nome,nome);
    temp->gradi = valC;
    temp->umidita = valH;
    temp->next = NULL;

    return lista;
}

ptrNode rimuoviLettture(ptrNode lista, char daOra[], char aOra[]){
    ptrNode temp;
    if (lista==NULL)
        return lista;
    if ((strcmp(lista->orario, daOra)>=0) && (strcmp(lista->orario,aOra)<=0)){
        temp = lista;
        lista = lista->next;
        free(temp);
        lista = rimuoviLettture(lista, daOra, aOra);
    }
    else{
        lista->next = rimuoviLettture(lista->next, daOra, aOra);
    }
    return lista;
}

ptrNode eliminaLista(ptrNode lista, int n, char *nome){
    ptrNode temp;

    if (lista==NULL)
        return lista;
    if ((n>0) && (strcmp(lista->nome,nome)==0)){
        temp = lista;
        lista = lista->next;
        free(temp);
        lista = eliminaLista(lista, n-1, nome);
    }
    else
        lista->next = eliminaLista(lista->next, n, nome);
    return lista;
}

void stampaLista(ptrNode lista){
    if (lista==NULL)
        return;
    printf("%s %s %.2f %.2f\n",lista->nome, lista->orario, lista->gradi, lista->umidita);
    stampaLista(lista->next);
}

```