Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria

Informatica A - a.a 2018/2019 - 5 Luglio 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	**6 punti**	_____
Esercizio 4	**6 punti**	_____
Esercizio 5	**10 punti**	_____

Totale(28) _____

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 C and not (A or B and not C)

A	**B**	**C**	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Risposta:
Semplificata: (a and b) or (not a and c)

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

(b) Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i numeri A = 46dec e B = -17, 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:
46d = 101110b- 6bit + 1 di segno = 7bit
cp2 = 0101110b
-17d = 10001b - 5 bit + 1 segno = 6bit
cp2 = 1101111b - cp2 7bit

A + B

riporto	11 111
A +	0101110b
B	1101111b
A+B	0011101b

Riporto perduto ma non overflow

A - B

riporto	
A -	0101110b
B	0010001b
A-B	0111111b

Né riporto perduto, né overflow

(c) Si converta il numero 28.6875 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:
5bit + segno
28d = 011100b
0.55 + 0.125 + 0.0625 = 0.1011
37.59375 = 011100.1011 ¡- Virgola fissa

segno = 0
mantissa = 1.11001011 non normalizzata
exp = 4 + 127 = 131d = 10000011

IEEE754
segno (1bit) = 0
esponente (8bit) = 10000011
mantissa (23bit) = 110 0101 1000 0000 0000 0000
elimino il primo 1

Esercizio 2 - Domanda di teoria (3 punti)

Illustrre le caratteristiche principali della memoria del calcolatore. In particolare spiegare:

1. La differenza tra memoria centrale e memoria di massa

2. Che cosa sono la memoria RAM e la memoria ROM; che ruolo hanno in un sistema di elaborazione, che cosa le differenzia.

Esercizio 3 - Esercizio C (6 punti)

Si scriva un sottoprogramma in linguaggio C che riceve in ingresso un array bidimensionale che contiene N espressioni, ognuna composta da numeri, parentesi e operatori +, -, x, /.
N viene passato alla funzione.
Il sottoprogramma deve salvare in un array bidimensionale di destinazione solamente le espressioni corrette, ovvero con le parentesi riportate correttamente (**numero di parentesi aperte uguale a numero di parentesi chiuse**). Per semplicita si assuma che le priorita tra le parentesi sia corrette (quindi non vi saranno espressioni del tipo "[2+(1x3])").
Il sottoprogramma deve anche restituire il numero di espressioni errate.

(Nell'esempio la funzione restituisce 1)

Array partenza:	Array destinazione:
[(3-2)x4+3]x(2+4)+(3-6)/4	[(3-2)x4+3]x(2+4) + (3-6)/4
[(3-2]x4+3]x(2+4))+(3-6)/4	5+6-[8x3]
5+6-[8x3]	7x8-(3x2)
7x8-(3x2)	

Si scriva inoltre il prototipo della funzione e la porzione di programma con l'invocazione della funzione (non è necessario scrivere tutto il main, scanf ecc.).

Risposta:

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

int modificaArray(char array[200][200], char arrayDest[200][200], int len){
    int r,q,g;

    int count = 0;

    int i;
    int j;
    for (i=0; i<len; i++)
    {
        r = 0;
        q = 0;
        g = 0;

        for (j=0; array[i][j]!='\0'; j++)
        {
            switch (array[i][j])
            {
                case '(':
                    r=r+1;
                    break;
                case ')':
                    r=r-1;
                    break;
                case '[':
                    q=q+1;
                    break;
                case ']':
                    q=q-1;
                    break;
                case '{':
                    g=g+1;
                    break;
```

```
                case '}':
                    g=g-1;
                    break;
            }
            if ((g<0) || (q<0) || (r<0)){
                count = count+1;
                break;
            }
        }
        if ((g==0) && (q==0) && (r==0))
            strcpy(arrayDest[i-count],array[i]);

    }
    return count;
}

void printArray(char array[200][200], int len){
    for (int i=0; i<len; i++)
        printf("%s",array[i]);
}

int main(int argc, const char * argv[]) {
    char array[200][200];
    char arrayDest[200][200];
    int k;

    strcpy(array[0],"{[(3-2)x4+3]x(2+4)}+(3-6)/4\n");
    strcpy(array[1],"{[(3-2]x4+3]x(2+4))+(3-6)/4\n");
    strcpy(array[2],"5+6-[8x3]\n");
    strcpy(array[3],"7x8-(3x2)\n");

    printArray(array, 4);

    k = modificaArray(array, arrayDest, 4);

    printf("\n\nArray destinazione: \n");
    printArray(arrayDest, 4-k);
    printf("\n\nEspressioni errate: %d\n",k);
    return 0;
}
```

Esercizio 4 - Matlab (6 punti)

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

(a) Scrivere una funzione che presi in ingresso il numero di righe ed il numero di colonne, crea la matrice M(righe,colonne) contenente i primi N = (righe x colonne) numeri dispari(max 12 riga) (2 punto)

Risposta:

```
function out = luglio5_19(row,column)
    c = 1;
    for (i=1:row)
        for (j=1:column)
            out(i,j) = c;
            c = c+2
        end
    end
end
```

(b) Sostituire in una colonna a caso, tutti valori pari a uno (NB.: l'istruzione deve essere la più generale possiible, in modo da poter essere utilizzata con matrici di dimensini differenti) (max 1 riga) (1 punto).

Risposta:

```
A(:,randi(size(A,2))) = ones(size(A,1),1)
A(:,randi(size(A,2))) = ones(1,size(A,1))
```

(c) Eliminare la riga con media più bassa (max 2 riga) (1 punto).

Risposta:

```
M(mean(M,2)==min(mean(M,2)),:)=[]
```

(d) Moltiplicare tutte le celle per un numero casuale intero tra 5 e 18 (max 1 riga) (1 punto).

Risposta:

```
M = M * randi([5,18],1)
M = M * randi([5,18])
```

(e) Moltiplicare per -1 tutte le celle che contengono numeri divisibili per 7 (max 1 riga) (1 punto).

Risposta:

```
M(rem(M,7)==0) = M(rem(M,7)==0)*-1
M(mod(M,7)==0) = M(mod(M,7)==0)*-1
```

Esercizio 5 - Programmazione C Liste (10 punti)

Si vuole creare un programma per la gestione di software. Tutte i software disponibili sono contenuti in **una lista**. Ogni nodo della lista corrisponde ad un software ed ha come attributi:

- nome software
- versione
- software house
- prezzo

es.

- nome software: Winzoz9
- versione: 5
- software house: Microfrost
- prezzo: 10

1. Definire la struttura dati necessaria per la realizzazione del programma. (1 punto)

2. Scrivere la funzione **RICORSIVA** che ricerca un software all'interno della lista. La funzione riceve in ingresso il nome del software e la versione e:
   - Se il nome del software non è presente nella lista dei software, la funzione restituisce -1;
   - Se il nome del software è presente, ma non lo è la versione, restituisce -2.
   - Se software e versione sono presenti, restituisce 0 e il node contente il software.

   (3 punti)

3. Utilizzando la funzione precedente, scrivere una funzione che presi in ingresso la lista completa dei software e due array contenenti nome software e versione (lunghezza massima dell'array 10 - alla funzione va passata la lunghezza dell'array), crea una nuova lista con i software elencati nell'array. (3 punti)

4. Scrivere una funzione **RICORSIVA** che presi in ingresso la lista e il nome di un software, ritorni quante versioni del software sono state rilasciate.

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

Risposta:

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

//Punto 1 ----------------------
typedef struct nodo{
    char nomeSoftware[20];
    int versione;
    char softwareHouse[20];
    int prezzo;
    struct nodo *next;
} nodo;

typedef nodo *ptrNode;
//-----------------------------
```

```
void stampaLista(ptrNode lista)
{
    if (lista == NULL)
        return;
    printf("Nome software: %s\nVersione;: %d\nSoftware house: %s\nPrezzo: %d\n\n",
    lista->nomeSoftware,lista->versione,lista->softwareHouse,lista->prezzo);
    stampaLista(lista->next);
}

ptrNode inserisci(ptrNode lista, char nomeSoftware[20], int versione, char softwareHouse[20],
int prezzo)
{
    ptrNode nuovo;
    nuovo = (ptrNode)malloc(sizeof(nodo));

    strcpy(nuovo->nomeSoftware, nomeSoftware);
    nuovo->versione=versione;
    strcpy(nuovo->softwareHouse,softwareHouse);
    nuovo->prezzo = prezzo;

    nuovo->next = lista;
    return nuovo;
}

//Punto 2: restituisce un nodo nuovo, contenente i dati del softwre ricercato.
//La malloc viene fatta nel main.
int ricerca(ptrNode lista, ptrNode nodoTrovato, int nome, char nomeSoftware[20], int versione)
{
    if (lista == NULL){
        nodoTrovato = NULL;
        if (nome==0)
            return -1;
        else
            return -2;
    }
    if (strcmp(lista->nomeSoftware,nomeSoftware)==0) //trovato nome almeno una volta
    {
        if (lista->versione == versione) //trovato software corretto
        {
            strcpy(nodoTrovato->nomeSoftware, lista->nomeSoftware);
            nodoTrovato->versione = lista->versione;
            strcpy(nodoTrovato->softwareHouse, lista->softwareHouse);
            nodoTrovato->prezzo = lista->prezzo;
            nodoTrovato->next = NULL;
            return 0;
        }
        else
            return ricerca(lista->next,nodoTrovato,1,nomeSoftware,versione);
    }
    else
        return ricerca(lista->next,nodoTrovato,0,nomeSoftware,versione);

}

//Punto 2: restituisce il puntatore al software ricercato. Non pu essere usato direttametne per
//creare la nuova lista, altrimenti si modificherbbe anche la lista di partenza.
int ricerca2(ptrNode lista, ptrNode *nodoTrovato, int nome, char nomeSoftware[20], int versione)
{
    if (lista == NULL){
        *nodoTrovato = NULL;
```

```
        if (nome==0)
            return -1;
        else
            return -2;
    }
    if (strcmp(lista->nomeSoftware,nomeSoftware)==0) //trovato nome almeno una volta
    {
        if (lista->versione == versione) //trovato software corretto
        {
            *nodoTrovato = lista;
            return 0;
        }
        else
            return ricerca2(lista->next,nodoTrovato,1,nomeSoftware,versione);
    }
    else
        return ricerca2(lista->next,nodoTrovato,0,nomeSoftware,versione);

}

//Punto 3, con funzione che restituisce un nuovo nodo, contente i dati del software trovato
ptrNode creaLista(ptrNode listaSoftware, char arrayNomi[10][20], int arrayVersioni[10], int len){
    int i = 0;
    ptrNode trovato;
    ptrNode lista = NULL;


    if (listaSoftware == NULL)
        return NULL;

    for (i = 0; i<len; i++)
    {
        trovato = (ptrNode)malloc(sizeof(nodo));
        if (ricerca(listaSoftware, trovato, 0, arrayNomi[i], arrayVersioni[i])==0)
        {
            trovato->next = lista;
            lista = trovato;
        }
    }
    return lista;
}

//Punto 3, con funzione che restituisce il puntatore al software trovato.
ptrNode creaLista2(ptrNode listaSoftware, char arrayNomi[10][20], int arrayVersioni[10], int len){
    int i = 0;
    ptrNode trovato = NULL;
    ptrNode lista = NULL;
    ptrNode temp = NULL;

    if (listaSoftware == NULL)
    return NULL;

    for (i = 0; i<len; i++)
    {
        if (ricerca2(listaSoftware, &trovato, 0, arrayNomi[i], arrayVersioni[i])==0)
        {
            temp = (ptrNode)malloc(sizeof(nodo));
            strcpy(temp->nomeSoftware,trovato->nomeSoftware);
            temp->versione = trovato->versione;
            temp->prezzo = trovato->prezzo;
```

```
            strcpy(temp->softwareHouse,trovato->softwareHouse);
            temp->next = lista;
            lista = temp;
        }
    }
    return lista;
}

int contaVersioni(ptrNode listaSoftware, char nomeSoftware[20])
{
    if (listaSoftware == NULL)
        return 0;
    else
    {
        if (strcmp(listaSoftware->nomeSoftware,nomeSoftware)==0)
            return 1+contaVersioni(listaSoftware->next, nomeSoftware);
        else
            return contaVersioni(listaSoftware->next, nomeSoftware);
    }
}


int main(int argc, const char * argv[]) {
    ptrNode listaSoftware = NULL;
    ptrNode mysoftware = NULL;

    ptrNode trovato;

    char nomeSoftware[20];
    int versione;

    char softwares[10][20];
    int versions[10];

    strcpy(softwares[0],"winzoz9");
    strcpy(softwares[1],"winzoz9");
    strcpy(softwares[2],"winzoz9");
    strcpy(softwares[3],"winzoz9");
    versions[0] = 3;
    versions[1] = 1;
    versions[2] = 5;
    versions[3] = 11;

    int r;
    int k;

    listaSoftware = inserisci(listaSoftware,"winzoz9",3,"Microfrost",15);
    listaSoftware = inserisci(listaSoftware,"winzoz9",2,"Microfrost",10);
    listaSoftware = inserisci(listaSoftware,"winzoz9",1,"Microfrost",10);
    listaSoftware = inserisci(listaSoftware,"winzoz9",4,"Microfrost",10);
    listaSoftware = inserisci(listaSoftware,"maxos",12,"Pineapple",10);
    listaSoftware = inserisci(listaSoftware,"winzoz9",5,"Microfrost",100);
    listaSoftware = inserisci(listaSoftware,"maxos",10,"Pineapple",0);
    listaSoftware = inserisci(listaSoftware,"maxos",5,"Pineapple",100);

    stampaLista(listaSoftware);

    do{
        printf("MENU\n");
        printf("1) Ricerca\n");
```

```
        printf("2) Ricerca con puntatore\n");
        printf("3) Crea nuova lista\n");
        printf("4) Conta versioni\n");
        printf("5) ESCI\n\n>> \n");
        scanf("%d",&r);
        switch (r){
            case 1:
                //RICERCA
                printf("Nome software: ");
                scanf("%s",nomeSoftware);
                printf("Versione: ");
                scanf("%d",&versione);

                trovato = (ptrNode)malloc(sizeof(nodo));

                k = ricerca(listaSoftware, trovato, 0, nomeSoftware, versione);
                if (k==0)
                {
                    printf("Software trovato\n\n");
                     printf("Nome software: %s\nVersione;: %d\nSoftware house: %s\nPrezzo: %d\n\n",
                     trovato->nomeSoftware,trovato->versione,trovato->softwareHouse,trovato->prezzo);
                }
                else if (k==-1)
                    printf("Software non nella lista\n\n");
                else
                    printf("Esiste un'altra versione del software ma non quella ricercata\n\n");
                break;

            case 2:
                printf("Nome software: ");
                scanf("%s",nomeSoftware);
                printf("Versione: ");
                scanf("%d",&versione);
                k = ricerca2(listaSoftware, &trovato, 0, nomeSoftware, versione);
                if (k==0)
                {
                    printf("Software trovato\n\n");
                    printf("Nome software: %s\nVersione: %d\nSoftware house: %s\nPrezzo: %d\n\n",
                    trovato->nomeSoftware,trovato->versione,trovato->softwareHouse,trovato->prezzo);
                }
                else if (k==-1)
                printf("Software non nella lista\n\n");
                else
                printf("Esiste un'altra versione del software ma non quella ricercata\n\n");
                break;

            case 3:
                //CREA LISTA
                mysoftware = creaLista(listaSoftware, softwares, versions, 4);
                //mysoftware = creaLista2(listaSoftware, softwares, versions, 4);
                stampaLista(mysoftware);
                break;

            case 4:
                //CONTA VERSIONI
                printf("Numero di versioni di Winzoz9: %d\n",contaVersioni(listaSoftware, "winzoz9"));
                break;

        }
    }while(r!=5);
```

```
}
```