Cognome ..... Nome ..... Matricola ..... CFU ....

Programmazione - Appello del 12/9/2022

Istruzioni

- Prima di iniziare, scrivete nome, cognome e matricola in alto su *ogni foglio*. Indicate il numero di crediti del vostro esame nel campo CFU.
- Ogni esercizio riporta la scritta "completare se CFU $\geq$ x": dovete completare *solamente* gli esercizi per cui i crediti del vostro esame sono almeno x.
- Dall'inizio della prova, per consegnare il compito alla cattedra, avete a disposizione un tempo che dipende dai CFU del vostro esame e se avete diritto al 30% di tempo aggiuntivo, secondo la seguente tabella:

CFU	Minuti	+30%
3	45	59
6-7	70	91
9-10	90	117

- Potete usare solamente penne, matita e gomma. Nient'altro può essere presente sul banco. Non potete usare fogli aggiuntivi, nemmeno per la brutta copia. Appoggiate zaini e giacche a lato della stanza. Scrivete in modo leggibile, parti non comprensibili potranno non essere corrette.

Esercizio 1 **4 punti** (completare se **CFU** $\geq$ **9**)

Scrivere le istruzioni per effettuare le seguenti dichiarazioni:

(a) una matrice m di reali a doppia precisione di 5 righe e 2 colonne — 1 punto

(b) un vettore v di 10 elementi, ciascuno dei quali è un puntatore a float. — 1 punto

(c) un puntatore p ad un vettore di 10 interi positivi — 2 punti

Risposta:

```
1    (a) double m[5][2];
2    (b) float *v[10];
3    (c) unsigned int (*p)[10];
```

Esercizio 2 **4 punti** (completare se **CFU** $\geq$ **9**)

Specificare se il codice seguente compila e, nel caso, indicare cosa stampa alle righe 11, 14 e 15. Motivare brevemente le risposte.

```
1  #include <stdio.h>

3  int m[2][3]={1,2,3,4,5,6};
4  int x = 2;

6  int main(void) {

8      int *p = (int *) m;
9      {
10         int x = 1;
11         printf("%d\n", *(p+x));
12     }
13     int **q = &p;
14     printf("%d\n", *(*q+x));
15     printf("%d\n", *(*(m+1)+1));
16 }
```

Risposta:

Il codice compila e stampa
2 (riga 11)
3 (riga 14)
5 (riga 15)
Le I_n indicano indirizzi di memoria, gli apici la riga dove la variabile è stata definita, ~~x~~ indica che la variabile x non è visibile in quella riga.

riga 10		
I_1	$m[0][0]^3$	1
I_1+1	$m[0][1]^3$	2
I_1+2	$m[0][2]^3$	3
I_1+3	$m[1][0]^3$	4
I_1+4	$m[1][1]^3$	5
I_1+5	$m[1][2]^3$	6
I_2	~~x^4~~	2
I_3	p^8	I_1
I_4	x^{10}	1

$\rightarrow$ $*(p^8+x^{10})=*(I_1+1)=*(\&m[0][1])=2;$

riga 13		
I_1	$m[0][0]^3$	1
I_1+1	$m[0][1]^3$	2
I_1+2	$m[0][2]^3$	3
I_1+3	$m[1][0]^3$	4
I_1+4	$m[1][1]^3$	5
I_1+5	$m[1][2]^3$	6
I_2	x^4	2
I_3	p^8	I_1
I_5	q^{13}	I_3

$\rightarrow$

$\rightarrow$ $*q=I_1=\&m[0][0]$, $*(*q+x^4)=*(\&m[0][2])=3$; $*(m+1)=\&m[1][0]$ e $*(*(m+1)+1)=*(\&m[1][1])=5$

Cognome ..... Nome ..... Matricola ..... CFU ....

Esercizio 3 **8 punti** (completare se **CFU** $\geq$ **6**)

Definire una funzione ricorsiva che calcoli il numero di percorsi che permettono di attraversare un campo fiorito, dal basso verso l'alto, senza calpestare alcun fiore. Il campo è rappresentato da una matrice le cui dimensioni sono indicate da due costanti DIM_X e DIM_Y. Un valore della matrice rappresenta la presenza di un fiore (0) oppure la sua assenza (1) in tale posizione. E' possibile muoversi una casella in alto oppure una casella verso destra. Ad esempio se il campo è definito come segue:

```
{0,0,0,1,0}
{0,1,0,1,1}
{1,0,1,1,0}
{1,0,1,1,1}
{1,0,1,0,0}
```

e la casella di partenza è (4,2), esistono due percorsi (per cui la funzione dovrà restituire 2), raffigurati dalle X :

```
{0,0,0,X,0}        {0,0,0,X,0}
{0,1,0,X,1}        {0,1,0,X,1}
{1,0,X,X,0}        {1,0,1,X,0}
{1,0,X,1,1}        {1,0,X,X,1}
{1,0,X,0,0}        {1,0,X,0,0}
```

Risposta:

```
1  #define DIM_X 5
2  #define DIM_Y 5

4  int mossa(int campo[DIM_X][DIM_Y], int pos_x, int pos_y) {
5      /*
6          PRE: campo ha dimensioni DIM_X e DIM_Y.
7          POST: restituisce il numero di percorsi che partono da
8          (pos_x, pos_y) e giungono alla riga 0, evitando le celle
               uguali a 0.
9      */

11     if ((pos_x<0) || (pos_x>DIM_X-1) || (pos_y<0) || (pos_y>DIM_Y-1))
12         return 0;
13     if (campo[pos_x][pos_y]==0)
14         return 0;
15     if (pos_x==0)
16         return 1;
17     return mossa(campo, pos_x-1, pos_y)+mossa(campo, pos_x, pos_y+1);
18 }
```

Esercizio 4 **6 punti** (completare se **CFU** $\geq$ **3**)

Tenendo conto delle dichiarazioni e del frammento di codice riportato di seguito, scrivere la funzione **iterativa** *first_odd* che, ricevuta una lista collegata con puntatori, dovrà restituire il puntatore al minimo elemento nella lista il cui valore è dispari (NULL se la lista è vuota o non contiene elementi dispari). Vi è richiesto di scrivere soltanto la funzione ed, eventualmente, le funzioni di appoggio definite/utilizzate.

```
1  #include <stdio.h>
2  #include <stdlib.h>

4  struct node {
5      int value;
6      struct node *nextPtr;
7  };

9  typedef struct node Lista;

11 Lista* first_odd(Lista *srcPtr);
```

Risposta:

Esercizio 5 **3 punti** (completare se **CFU** $\geq$ **3**)

Scrivere adesso la soluzione equivalente a quanto svolto nel precedente esercizio ma in forma **ricorsiva**.

Risposta:

Cognome ..... Nome ..... Matricola ..... CFU ....

Esercizio 6 **7 punti** (completare se **CFU** $\geq$ **3**)

Tenendo conto delle dichiarazioni e del frammento di codice riportato di seguito, scrivere la funzione *delete_node* che, dato un albero binario di ricerca, dovrà effettuare la cancellazione di un nodo mantenendo l'ordine e garantendo che l'albero sia effettivamente un albero binario di ricerca. Nota bene: si assume che la variabile ptrPtr punti "direttamente" al nodo da cancellare. Vi è richiesto di scrivere soltanto la funzione ed, eventualmente, le funzioni di appoggio definite/utilizzate.

```
1  #include <stdio.h>
2  #include <stdlib.h>

4  struct btree {
5      int value;
6      struct btree *leftPtr;
7      struct btree *rightPtr;
8  };

10 typedef struct btree BTree;

12 void delete_node(BTree **ptrPtr);
```

Risposta: