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

Programmazione - Appello del 29/8/2022

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**)

Specificare se il codice seguente compila e rispondere alle seguenti domande motivando brevemente ciascuna risposta:

(a) Cosa è possibile dire della variabile p alla riga 11? 2 punti

(b) Che valore viene stampato alla riga 15? 1 punto

(c) Che valore viene stampato alla riga 16? 1 punto

```
1  #include <stdio.h>

3  int main(void) {

5      int x=1, y=4;
6      int *p, *q=&x;
7      {
8          int x = 3;
9          p = &x;
10     }
11     printf("%d\n", *p);
12     p=q;
13     q=&y;
14     *q+=1;
15     printf("%d\n", y);
16     printf("%d\n", *p);
17 }
```

Risposta:

(a) p è una dangling reference alla riga 11; (b) 5; (c) 1.

Usiamo di seguito $I_1,\ldots,I_N$ come gli l-valori di delle variabili che definiamo. Di seguito x_k significa la variabile x dichiarata alla riga k, mentre il testo barrato significa una variabile non accessibile in quel momento.

riga 6		
I_1	x_5	1
I_2	y_5	4
I_3	p_6	ind.
I_4	q_6	I_1

$\rightarrow$

riga 9		
I_5	x_8	3
I_1	~~x_5~~	1
I_2	y_5	4
I_3	p_6	I_5
I_4	q_6	I_1

$\rightarrow$

riga 11		
I_1	x_5	1
I_2	y_5	4
I_3	p_6	I_5
I_4	q_6	I_1

$\rightarrow$

riga 12		
I_1	x_5	1
I_2	y_5	4
I_3	p_6	I_1
I_4	q_6	I_1

$\rightarrow$

riga 14		
I_1	x_5	1
I_2	y_5	5
I_3	p_6	I_1
I_4	q_6	I_2

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

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

Data la seguente funzione, scrivere PRE e POST condizioni e dimostrarne la correttezza.

```
1  int f(int X[], int dim) {
2
3      if(dim==0)
4          return 1;
5      if(X[0]%2==0)
6          return f(X+1, dim-1);
7      else
8          return 0;
9  }
```

Risposta:

POST: Restituisce 1 se tutti gli elementi di X sono pari; 0 altrimenti
PRE: dim equivale al numero di elementi di X

La dimensione del problema è rappresentata da dim.

Caso base: se $dim = 0, f(X, dim) = 1$ ed è vero che tutti gli elementi di X sono pari.

Caso induttivo: assumendo che la POST sia verificata per f(X+1, dim-1), dimostriamo che sia verificata per f(X,dim): se f(X+1,dim-1)=0 f(X,dim) restiuisce 0 in ogni caso (verificando la POST); se f(X+1,dim-1)=1 f(X,dim) resituisce 1 se X[0] è pari, 0 altrimenti; in entrambi i casi la POST è verificata.

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

Implementare una funzione che, a partire da una posizione nella scacchiera, segni sulla stessa tutte le mosse possibili per un alfiere.

Nel gioco degli scacchi, in una mossa un alfiere può spostarsi di un qualsiasi numero di caselle in diagonale. La scacchiera è una matrice 8x8. Un elemento della matrice assume i valori 0 (l'alfiere non può raggiungere questa posizione) e 1 (l'alfiere può raggiungere questa posizione). Si suppone che l'alfiere sia l'unico pezzo sulla scacchiera. Implementare la funzione mossa_alfiere() -scegliete voi la lista dei parametri- che, data una posizione (x,y) sulla scacchiera, assegni 1 alle posizioni della scacchiera raggiungibili da un alfiere posizionato in (x,y) e 0 alle altre.

Per esempio invocando mossa_alfiere() due volte consecutive nello stesso main per le posizioni (x,y)=(2,3) e (-1, 2) dopo ciascuna invocazione della funzione la scacchiera assume i seguenti valori:

```
          0 1 0 0 0 1 0 0
          0 0 1 0 1 0 0 0
          0 0 0 1 0 0 0 0
(2,3)->   0 0 1 0 1 0 0 0
          0 1 0 0 0 1 0 0
          1 0 0 0 0 0 1 0
          0 0 0 0 0 0 0 1
          0 0 0 0 0 0 0 0

          0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0
(-1,2)->  0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0
```

Risposta:

```
1  void inizializza(int m[][8]) {

3      for (int i=0; i<8; i+=1)
4          for (int j=0; j<8; j+=1)
5              m[i][j] = 0;

7  }

9  int mossa(int m[][8], int x, int y){

11     if (x>=0 && x<8 && y>=0 && y<8) {
12         m[x][y] = 1;
13         return 1;
14     }
15     return 0;
16 }

18 void mossa_alfiere(int m[][8], int x, int y) {

20     int i;
21     inizializza(m);
22     for(i=0; mossa(m, x+i, y+i); i+=1);
23     for(i=0; mossa(m, x-i, y-i); i+=1);
24     for(i=0; mossa(m, x+i, y-i); i+=1);
25     for(i=0; mossa(m, x-i, y+i); i+=1);
26 }
```

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

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

Tenendo conto delle dichiarazioni e del frammento di codice riportato di seguito, scrivere la funzione **ricorsiva** *clone_list* che, ricevuta una lista collegata con puntatori, dovrà crearne una copia / clone. Vi è richiesto di scrivere soltanto la funzione.

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

4  struct nodo {
5      float value;
6      struct nodo *nextPtr;
7  };

9  typedef struct nodo Lista;

11 void clone_list(Lista *srcPtr, Lista **destPtr);
```

Risposta:

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

Scrivere in modo chiaro ed esaustivo la definizione di albero binario di ricerca. Si riporti inoltre un esempio concreto che soddisfi tale definizione.

Risposta:

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

Tenendo conto delle dichiarazioni e del frammento di codice riportato di seguito, scrivere la funzione **ricorsiva** *ord_insert* che, dato un albero binario di ricerca, dovrà effettuare l'inserimento di nuovi nodi mantenendo l'ordine e garantendo che l'abero sia effettivamente un albero binario di ricerca. Vi è richiesto di scrivere soltanto la funzione.

```
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 ord_insert(BTree **ptrPtr, int val);
```

Risposta: