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

Programmazione - Appello del 1/2/2023

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 (se proprio ne avete bisogno richiedeteli al docente). 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, nel caso, commentare il codice alle righe 18, 20 e 21 indicando, ove possibile, cosa stampi. Motivare brevemente le risposte.

```
1  #include <stdio.h>

3  int * f(int x) {
4      int a[3] = {3,4,5};
5      a[x]+=x;
6      return a;
7  }

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

12 int main(void) {

14     int *p = (int *) m;
15     {
16         int x = 1;
17         int *h = f(x);
18         printf("%d\n", h[1]);
19     }
20     printf("%d\n", *(p+x));
21     printf("%d\n", *(*(m+2)+1));
22 }
```

Risposta:

Il codice compila (sperabilmente il compilatore fornisce un warning dovuto al fatto che a viene utilizzata fuori dalla funzione dove è definita).
Riga 18: h è una **dangling reference** per cui viene stampato un valore indefinito.
Riga 20: viene stampato **3**
Riga 21: viene stampato **6**.

Riga 20: p punta al primo elemento di m (*p==1), x ha valore due (la variabile definita alla riga 16 è stata deallocata quando si arriva alla riga 20), quindi *(p+2)=3

Riga 21:

$$m = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

con m+2 saltiamo alla terza riga (5 6); *(m+2)+1 seleziona il secondo elemento della terza riga.

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==1)
6          return f(X+1, dim-1);
7      else
8          return 0;
9  }
```

Risposta:

POST: Restituisce 1 se tutti gli elementi di X sono dispari; 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 dispari.

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 dispari, 0 altrimenti; in entrambi i casi la POST è verificata.

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

Scrivere una funzione rimuovi_non_ordinati() che scorra un array di interi eliminando alcuni suoi elementi in modo che, al termine della procedura, l'array risulti ordinato in modo crescente (per ogni i. $A[i] <= A[i+1]$). Il tipo restituito dalla funzione è void. I parametri formali della funzione sono l'array A contenente gli interi e la sua dimensione dim. Dopo l'invocazione della funzione, i parametri attuali corrispondenti ad A e dim (di seguito uso gli stessi nomi per semplicità) devono rispettare le seguenti condizioni:

- dim rappresenta il numero di elementi ordinati dell'array A.
- A deve contenere nelle prime dim posizioni i valori ordinati.

Risposta:

```
1  /*
2      PRE: dim equivale al numero di elementi dell'array A
3      POST: sposta tutti gli elementi per cui A[i]<A[i-1] alla fine
           dell'array; *dim equivale al numero di elementi ordinati
4  */
5  void rimuovi_non_ordinati(int *A, int *dim) {

7      int i,j;
8      if (*dim<=1)
9          return;
10     for(i=1; i<*dim; i++) {
11         if ( (A[i]<A[i-1]) && (i<*dim-1) ) {
12             // A[i] viola il vincolo, lo elimino traslando tutti gli
                   elementi alla sua destra di una casella verso
                   sinistra
13             for(j=i;j<*dim;j++)
14                 A[j] = A[j+1];
15             *dim -= 1;
16             i-=1;
17         }
18     }
19 }
```

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

Programmazione - Appello del 1/2/2023 - II PARTE

Esercizio 4 **7 punti** (completare se **CFU** $\geq$ **3**)

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

```
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/illustri inoltre un esempio concreto che soddisfi tale definizione.

Risposta:

Esercizio 6 **1 punto** (completare se **CFU** $\geq$ **3**)

Completare il seguente codice (la parte commentata) in modo che sia stampato a schermo il valore 3.

```
1  #include <stdio.h>

3  enum appelli{Gen, Feb, Giu, Lug, Set};

5  int main(void) {

7      // da completare (scrivere nel campo testo risposta)

9      return 0;
10 }
```

Risposta:

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

Esercizio 7 **5 punti** (completare se **CFU** $\geq$ **3**)

Tenendo conto delle dichiarazioni e del frammento di codice riportato di seguito, scrivere la funzione **ricorsiva** *search* che, dato un albero binario di ricerca, dovrà ricercare al suo interno un particolare valore *val*. La funzione restituirà 1 se l'elemento è presente. Vi è richiesto di scrivere soltanto la funzione.

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

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

10 typedef struct btree BTree;

12 int search(BTree *ptr, int val);
```

Risposta: