Programmazione - Compito del 22/4/2022

Istruzioni

- prima di iniziare, scrivete nome, cognome e matricola su *ogni foglio*. Se state sostendo un'integrazione, indicatelo sotto la matricola (es. integrazione 6 crediti).
- Dall'inizio della prova avete a disposizione un'ora (per quelli di voi che hanno diritto al 30% di tempo aggiuntivo 1:18h, 30 minuti per chi fa l'integrazione da 6-7 crediti) per consegnare il compito alla cattedra. Consegne in ritardo non verranno accettate. Una volta consegnato, non potete uscire dall'aula fino al termine del turno. Se state facendo un'integrazione, dovete completare solamente gli esercizi con indicato (integrazione $\geq$ x crediti).
- 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 non verranno corrette.
- Vi chiedo di non parlare e non alzare lo sguardo dal vostro foglio. Visto che siete seduti vicini gli uni agli altri, sarò intransigente, segnandomi la minima trasgressione e riservandomi successivamente come agire.

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

1. Discutere brevemente le differenze tra variabili globali, statiche e locali. **3 punti**

Risposta: (vedi diapositiva 4, lezione 12)
Le variabili globali sono create all'inizio dell'esecuzione, mantengono il loro valore e sono accessibili per tutta l'esecuzione.
Le variabili locali sono create quando il blocco dove sono definite viene eseguito, mantengono il loro valore finché si esegue il blocco e sono accessibili all'interno dello stesso blocco.
Le variabili statiche sono create all'inizio dell'esecuzione, mantengono il loro valore per tutta l'esecuzione e sono accessibili all'interno del blocco dove sono definite.

2. Se compiliamo ed eseguiamo il seguente codice, qual è il risultato (cosa viene stampato se non ci sono errori)? Motivare la risposta. **4 punti** (integrazione $\geq$ 6 crediti)

```
1  #include <stdio.h>
2
3  int x = 10;
4
5  int main(void) {
6      int y = 2;
7      {
8          int y = 4;
9          if (y>2)
10         {
11             int x = 2;
12             x += 3;
13         }
14         printf("%d", x/y*2);
15     }
16 }
```

Risposta: il codice stampa 4.
Nella riga 14 la variabile x è uguale a 10 perché la x definita alla riga 11 è stata deallocata. Si calcola prima 10/4=2 (la divisione tra interi restituisce il quoziente della divisione) che poi andiamo a moltiplicare per 2, ottenendo 4.

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

3. Utilizzando l'aritmetica dei puntatori, scrivere la formula per ottenere l'*indirizzo* dell'elemento `C[2][1][3]` nell'array seguente: **4 punti** (integrazione $\geq$ 6 crediti)

```
int C[3][4][5];
```

Risposta: C+(2*4*5)+(1*5)+3
Se volete provarlo al calcolatore C dovrebbe essere un puntatore ad intero, quindi ad esempio si può eseguire un cast (int *) C

4. Scrivere una funzione che, dato un array `A` di `dim` interi, calcoli la somma dei prodotti tra il primo e l'ultimo elemento dell'array, tra il secondo ed il penultimo ecc...Per esempio, dato l'array [1,2,3,4,5,6] la funzione calcola 1*6+2*5+3*4. Il tipo restituito dalla funzione è void, però deve essere possibile stampare nel main la somma calcolata. È richiesto di scrivere solamente la funzione. **8 punti**

```
1  void somma_prodotti_inversi_v1(int *X, int dim, int *somma) {
2
3      *somma = 0;
4      int i=0, j=dim-1;
5      while(i<=j) {
6          *somma += X[i] * X[j];
7          i+=1; j-=1;
8      }
9  }
```

Passando `somma` come puntatore, il valore calcolato all'interno della funzione può essere utilizzato fuori dalla funzione stessa. Si utilizzando due indici per scorrere l'array, uno da sinistra a destra (i) e l'altro da destra a sinistra (j). Notate che la condizione $i \leq j$ permette di considerare il caso di array pari e dispari. La versione seguente è equivalente, l'unica differenza è che usa un for invece del while.

```
1  void somma_prodotti_inversi_v2(int *X, int dim, int *somma) {
2
3      *somma = 0;
4      for(int i=0, j=dim-1;i<=j;i+=1,j-=1) {
5          *somma += X[i] * X[j];
6      }
7  }
```

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

5. Data la seguente funzione, scrivere PRE e POST e discuterne la correttezza. **3 punti** (integrazione $\geq$ 6 crediti)

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

Risposta:
PRE: $dim > 0$ è la dimensione dell'array X.
POST: restituisce il valore massimo in X.
La ricorsione è definita rispetto alla dimensione dell'array, ogni chiamata ricorsiva è fatta su un array di dimensione più piccola, fino a raggiungere il caso base di un array con un solo elemento (l'array vuoto è escluso dalla precondizione).
Caso base: se l'array ha un solo elemento, quell'elemento è anche il massimo
Caso induttivo: per ipotesi induttiva assumiamo che la POST valga per la chiamata alla riga 5, f(X+1, dim-1), quindi m è il massimo tra gli elementi dell'array X a partire dalla seconda all'ultima posizione. Alla riga 6 si restituisce il massimo tra il primo elemento di X ed m, dimostrando che la POST vale per f(X, dim).

6. Scrivere una funzione ricorsiva che, dati due array di caratteri contenenti ciascuno una stringa, copia il contenuto della seconda stringa nella prima. **8 punti** (integrazione $\geq$ 6 crediti)

```
1  void copia_stringa(char *s1, char *s2) {
2      /*
3          PRE: s1, s2 sono stringhe. L'array di caratteri contenente
4          s1 e' piu' lungo di s2.
5          POST: copia il contenuto di s2 in s1
6      */
7      *s1 = *s2;
8      if (*s2!='\0')
9          copia_stringa(s1+1, s2+1);
10 }
```