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

Programmazione
Compitino del 26/4/2023

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 (se vi siete iscritti negli ultimi 3-4 anni e non state sostenendo un'integrazione, il numero di crediti è con tutta probabilità 9).
- Ogni esercizio riporta la scritta "completare se CFU $\geq$ x": dovete completare *solamente* gli esercizi per cui i crediti (CFU) del vostro esame sono almeno x.
- Dall'inizio della prova, per consegnare il compito, avete a disposizione un tempo che dipende dai CFU del vostro esame, secondo la seguente tabella:

Minuti a disposizione

CFU	Sezione I	Sezione II
6-7	14	24
9-10	20	45

- Iniziate lavorando agli esercizi della sezione I su carta. In questa fase non potete usare il calcolatore.
- Al termine della prima sezione, dopo che abbiamo ritirato tutti i compiti, daremo il via alla seconda sezione e solo allora potrete accendere lo schermo del calcolatore e lavorare agli esercizi di programmazione sul Moodle Esercizi. Gli esercizi della seconda sezione sono da consegnare al calcolatore, possono essere consegnati su carta solo in caso di malfunzionamenteo prolungato del calcolatore. In questo caso dovete indicare sul compito che consegnerete che deve essere corretta la versione su carta (trovate una casella da marcare all'inizio della seconda sezione).
- Potete usare solamente penne, matita e gomma. Nient'altro può essere presente sul banco, deve però esserci un documento di riconoscimento. 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 **3 punti** (completare se **CFU** $\geq$ **9**)

Specificare se il codice seguente compila e, nel caso, commentare il codice alle righe 14, 16 e 18 indicando, ove possibile, cosa stampi. Motivare brevemente le risposte.

```
1  #include <stdio.h>

3  int x = 12;

5  int main() {
6      int y = 2;
7      int *p;
8      {
9          int y = 4;
10         if (y>2) {
11             int x = 2;
12             p = &x;
13             *p += 8;
14             printf("%d\n", x/4*2);
15         }
16         printf("%d %d\n", x, y);
17     }
18     printf("%d\n", *p);
19 }
```

Risposta:

Il codice compila;
alla riga 14 stampa 4;
alla riga 16 stampa 12 4;
alla riga 18 stampa un valore indefinito (dangling reference).

x^5 significa che la variabile x dichiarata alla riga 5, ~~x^5~~ significa che la variabile x non è visibile alla riga corrente. Vediamo come si configura la memoria dopo aver eseguite specifiche righe:

L-val.	id.	R-val.
	riga 12	
I_5	x^{11}	2
I_4	y^9	4
I_3	p^7	I_5
I_2	~~y^6~~	2
I_1	~~x^3~~	12

$\rightarrow$

L-val.	id.	R-val.
	riga 14	
I_5	x^{11}	10
I_4	y^9	4
I_3	p^7	I_5
I_2	~~y^6~~	2
I_1	~~x^3~~	12

$\rightarrow$

L-val.	id.	R-val.
	riga 16	
I_4	y^9	4
I_3	p^7	I_5
I_2	~~y^6~~	2
I_1	x^3	12

$\rightarrow$

L-val.	id.	R-val.
	riga 18	
I_3	p^7	I_5
I_2	y^6	2
I_1	x^3	12

alla riga 14 x=10, x/4=2 (divisione tra interi), 2*2=4. Alla riga 18 la locazione di memoria I_5 non è più accessibile, il programma non ha il controllo su quella cella di memoria, per cui il valore è indefinito.

Esercizio 2 **1 punto** (completare se **CFU** $\geq$ **6**)

Data la seguenti dichiarazioni

```
1          double x[5][3];
2          double *p = &x[0][0];
```

utilizzando l'aritmetica dei puntatori, completare l'espressione seguente per ottenere l'elemento x[4][2]: *(p + ...)

Risposta:

*(p+4*3+2)

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

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

Data la seguente funzione,

```
1 #include <stdio.h>
2
3 int f(int n) {
4     if(n==1)
5         return 1;
6     else
7         return 2*f(n-1);
8 }
```

(a) scrivere la sua POSTcondizione;

(b) scrivere la sua PREcondizione;

(c) dimostrarne la correttezza;

Risposta:

POST: Restituisce 2^{n-1}
PRE: $n \geq 1$

Caso base: se $n = 1$, la funzione restituisce 1; POST=$f(n) = 2^0 = 1$, quindi la POST è verificata.

Caso ricorsivo: assumendo che la POST sia verificata per f(n-1), ovvero che $f(n-1) = 2^{n-2}$, dimostriamo che sia verificata per f(n): la funzione restituisce $2 * f(n-1) = 2 * 2^{n-2} = 2^{n-1}$, quindi la POST è verificata.

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

Sezione II: Esercizi da Eseguire al Calcolatore

Link al Moodle:

LabP140 e LabP036: https://elearning.studenti.math.unipd.it/labs/course/view.php?id=25

LabP104: http://10.190.1.252

Segnate la casella a fianco se NON avete potuto consegnare gli esercizi seguenti su Moodle e volete che sia corretta la versione cartacea ☐

Esercizio 4 **5 punti** (completare se **CFU** $\geq$ **9**)

Scrivete una funzione min_max che trovi l'indice del minimo e del il massimo valore di un array di interi positivi passato come parametro.

Nel caso in cui il valore minimo non sia unico, si deve restituire quello di indice minore. Nel caso in cui il valore massimo non sia unico, si deve restituire quello di indice maggiore. Nel caso in cui non sia possibile trovare tali indici, il valore calcolato dalla funzione sarà (-1,-1).

Nel main deve essere possibile stampare gli indici calcolati dalla funzione. Ad esempio se A=3, 5, 1, 7, 0, 9, 4, 6, 2, 8 il codice nel main stampa "4 5" (indici di 0 e 9)

```
1   /*
2       PRE:
3       POST:
4   */
5   int main() {
6       int N;
7       scanf("%d", &N);
8       int array[N];
9       for(int i = 0; i < N; i++){
10          scanf("%d", array+i);
11      }
12      int min_indice, max_indice;

14      // INVOCATE QUA LA VOSTRA FUNZIONE min_max

16      printf("%d %d\n", min_indice, max_indice);
17  }
```

Risposta:

```
1  /* PRE: A ha dim elementi
2     POST: per ogni l.
3       0 <= l <*i: A[*i]<A[l], *i < l <dim: A[*i]<=A[l];
4       per ogni m.
5       *j < m < dim: A[*j]>A[m], per ogni m. 0 <= m < *j: A[*j]>=A[m]
6  */
7  void min_max(int *A, int dim, int *i, int *j) {

9      *i = 0, *j = 0;
10     for (int k = 0; k < dim; k += 1){
11         if (A[k] < A[*i])
12             *i = k;
13         if (A[k] >= A[*j])
14             *j = k;
15     }
16     if(dim==0) {
17         *i=-1;
18         *j=-1;
19     }
20 }
```

Esercizio 5 **3 punti** (completare se **CFU** $\geq$ **6**)

Definire una funzione ricorsiva che inverta una stringa. Il prototipo della funzione è:

void inverti_stringa(char *s, int dim);

dove dim è la dimensione della stringa. Ad esempio se inizialmente s="ciao", dopo l'invocazione della funzione s="oaic".

```c
#include <stdio.h>

int lung_string(char *s) {
    if(*s=='\0')
        return 0;
    return 1+lung_string(s+1);
}


int main(void) {

    char a[100];
    scanf("%s", a);
    inverti_stringa(a, lung_string(a));
    printf("%s\n", a);

}
```

Risposta:

```c
#include <stdio.h>

/*
    PRE: dim equivale alla lunghezza della stringa s
    POST: al termine s viene invertita
*/
void inverti_stringa(char *s, int dim) {

    char t;
    if(dim>1) {
        t = s[0];
        s[0] = s[dim-1];
        s[dim-1] = t;
        inverti_stringa(s+1, dim-2);
    }
}
```