TEST DI PROGRAMMAZIONE - 6 settembre 2016

È necessario rispondere correttamente ad almeno 7 domande su 10 affinché venga valutata la seconda parte di esercizi.

Cognome e Nome: _____

Matricola: _____

1. Data la seguente funzione:

```
let rec f x =
    match x with
    | [] -> 0
    | y::z -> if (y mod 2 <>0) then y + (f z) else (f z);;
```

Qual è il tipo della funzione ?

A. `int -> int list` B. `int -> int` C. `int list -> int` D. `(int -> int) -> int`

1. _____

2. Data la funzione `f` dell'esercizio precedente, dire che cosa restituisce.

A. la somma degli elementi in posizione pari di una lista di interi B. la somma degli elementi in posizione dispari di una lista di interi C. la somma degli elementi dispari di una lista di interi D. la somma degli elementi pari di una lista di interi

2. _____

3. Dire quale delle seguenti espressioni ha come valore `int list = [2;4;7;5;9]` :

A. `[2]::[4;7;5;9]` B. `[2;4;7]::[5;9]` C. `2@[4;7;5;9]` D. `[2;4;7]@[5;9]`

3. _____

4. Data la funzione

```
let rec g (n, x) =
    if n = 0 then []
    else x::g(n-1,x);;
```

dire quali delle seguenti chiamate di funzione è corretta.

A. `g(5, 'a')` B. `g([5], 'a')` C. `g([5], 1)` D. `g('a', 1)` E. `g('a', [1])` F. `g([1], [5])`

4. _____

5. Date le dichiarazioni `int a; float b,c,d;` quale fra i seguenti assegnamenti non è corretto a causa di un errore di tipo? A. `a = (b>c);` B. `a = (b+1 == c-1);` C. `&a = &d;` D. Sono tutti assegnamenti corretti

5. _____

6. Data la seguente funzione:

```
void somme(int a, int b) {
   int res = a + b;
   a = a + res;
}
```

e il seguente codice:

```
int k1=5, k2=3;
somma(k1, k2);
printf(''%d %d'', k1, k2);
```

Quale delle seguenti è corretta?

A. Il codice non compila B. Il programma stampa 8 5 C. Il programma stampa 8 3 D. Il programma stampa 5 3

6. _____

7. Nel linguaggio C, il passaggio dei parametri avviene...

A. Per copia B. Per copia e referenza C. Per referenza D. Per variabile

7. _____

8. Nel linguaggio C, i parametri effettivi (attuali) vengono valutati generalmente in...

A. compilazione B. esecuzione C. linking D. stesura

8. _____

9. Date le dichiarazioni `int a; int *b, int **c`, quale fra le seguenti risposte *non* può stare a sinistra di un assegnamento?: A. `*b` B. `*(b + a - (**c))` C. `*b + a` D. `*(b+2*a)`

9. _____

10. Dato il seguente ciclo:

```
int i,j;
int acc = 0;
for (i = 0; i < 2; i++)
   for (j=0; j < 2; j++)
      acc = acc + i*10 + j;
printf(''%d'', acc);
```

Cosa stampa?

A. 12 B. Non è corretto perché non inizializza le variabili `i,j` C. 3 D. 22

10. _____

i j ACC
0 0 0
0 1 1
1 0 11
1 1 11+10+1 = 22

ESAME DI PROGRAMMAZIONE
(Programmazione Funzionale)

Cognome:

Nome:

Matricola:

Data: **6/09/2016**

Svolgere i seguenti quesiti nel foglio di protocollo. Consegnare: presente testo, bella e brutta copia. Tempo previsto: 60 minuti.

1. **Esercizio 1**
   Scrivere una funzione `ordinati: 'a -> 'a list -> 'a list` che, dato un elemento `init` e una lista `input`, restituisca la lista `output` tale che:

   - il primo elemento della lista `output` è il primo elemento della lista `input` maggiore o uguale a `init` (se `input` contiene un tale elemento); gli elementi successivi della lista `output` si ottengono eliminando dalla lista `input` tutti quelli che non renderebbero la lista `output` ordinata. In altre parole, gli elementi della lista `output` compaiono nello stesso ordine in cui occorrono nella lista `input`, e ciascuno è maggiore o uguale al prededente.
   - Se la lista `input` non contiene alcun elemento maggiore o uguale a `init`, allora `output = []`.

   Ad esempio,

   ```
   ordinati 3 [1;2;3;0;5;4;8;7;10;9] = [3; 5; 8; 10]
   ```

2. **Esercizio 2**
   Scrivere una funzione `elimina_precedenti: 'a -> 'a list -> 'a list` che, applicata a un elemento `x` e una lista `lst` restituisca la lista che si ottiene da `lst` eliminando tutti gli elementi che precedono `x` in `lst`. La funzione restituirrà la lista vuota se `x` non compare in `lst`.

3. **Esercizio 3**
   Scrivere una funzione `sub_ord: 'a list -> 'a list list`, che, applicata a una lista di elementi (di un tipo con uguaglianza) `lst`, restituisca una lista di liste, ciascuna delle quali è ordinata (in senso non decrescente) e contiene il massimo numero possibile di elementi consecutivi di `lst`. Ciascun elemento di `lst` deve occorrere in una ed un'unica lista di `sub_ord lst`.
   Ad esempio:

   ```
   sub_ord [4;4;10;20;5;30;6;10] = [[4;4;10;20];[5;30];[6;10]]
   sub_ord [5;6;4;3;2;1] = [[5;6];[4];[3];[2];[1]]
   ```

ESAME DI PROGRAMMAZIONE
(Programmazione Imperativa)

Cognome:

Nome:

Matricola:

Data: **06/09/2016**

Svolgere i seguenti quesiti nel foglio protocollo. Consegnare: presente testo e bella copia. Barrare la parti che non si vuole vengano corrette e valutate. Tempo previsto: 90 minuti.

Svolgere gli esercizi di seguito riportati. Il candidato può introdurre, se lo ritiene, funzioni ausiliarie descrivendone l'utilità (a meno non sia specificato altrimenti):

In tutto l'esame useremo liste di interi dove ciascun intero può assumere il valore 0 oppure 1. La definizione sarà:

```
struct cella{
   int info;
   struct cella *next;
};

typedef struct cella *tlist;
```

1. **Esercizio 1**
   Implementare la funzione:

   ```
   void comprimi(tlist l, int **vett, int* size);
   ```

   che data una lista la comprima inserendo in un vettore un intero non nullo per ogni sequenza contigua di valori. L'intero n sarà strettamente positivo per indicare una sequenza di n 1 consecutivi, mentre sarà negativo per indicare una sequenza di $-n$ 0 consecutivi. Esempi:

   - La lista: 0-1-1-0-0-0-1-0 viene compressa nel vettore -1, 2, -3, 1, -1

   Il vettore va allocato in memoria dinamica della dimensione minima necessaria, e la sua dimensione va specificata tramite il parametro `size`. Si assuma che l'allocazione della memoria dinamica vada a buon fine. Non modificare la lista in ingresso. Si dia un main di prova.

2. **Esercizio 2**
   Implementare la funzione *ricorsiva*:

   ```
   unsigned int converti(tlist l);
   ```

   che, data la lista l restituisca l'intero associato alla codifica binaria contenuta nella lista. Si assuma che la testa della lista abbia il bit meno significativo e che la lista vuota codifichi lo zero.

3. **Esercizio 3**
   Si implementi la seguente funzione:

   ```
   tlista maxseq(tlista l);
   ```

   che restituisca il puntatore alla prima cella della sequenza di celle contigue uguali di lunghezza maggiore. In caso di sequenze di pari lunghezza si restituisca la cella più vicina alla testa della lista. Nei seguenti esempi l'elemento in grassetto sarà quello puntato dal risultato:

   - $1-0-0-\mathbf{1}-1-1-0-0-0$
   - $\mathbf{0}-1-0-1$
   - $0-0-1-\mathbf{0}-0-0-0-1$

   Se la lista in input è la lista vuota, restituire NULL.