Programmazione I
Prova scritta - 18 settembre 2013 - 1h20min

NOTA: Nei programmi si trascuri ogni problema legato al tipo ed al valore di ritorno della funzione **main**, inoltre si sottintenda la presenza delle direttive
`#include <iostream>` / `#include <fstream>` / `using namespace std ;`
e non si prenda come un buon esempio la formattazione utilizzata (spesso compressa per motivi di spazio). Si interpreti "terminazione forzata", come l'abbreviazione di "terminazione forzata del programma da parte del sistema operativo". Infine, laddove si trovi l'affermazione che un programma o frammento di codice produce un certo risultato, è da intendersi che, in accordo alle regole del linguaggio, tale programma o frammento di codice produce quel risultato per **qualsiasi esecuzione** su **qualsiasi macchina**.

PARTE 1 – RISPOSTA SINGOLA - Ogni domanda ha una sola risposta VERA.

- **Una risposta esatta fa acquisire il punteggio positivo riportato a fianco della domanda**
- **Una risposta errata fa perdere il punteggio negativo riportato a fianco della domanda**
- **Una risposta lasciata in bianco viene valutata 0**

1. **(3, -0.5)** Dato il seguente programma
```
main()
{
 int i = 10 ; char s[10] ; ifstream f("dati.txt");
 f>>i ; f>>s ;
 cout<<i<<" "<<s<<endl ;
}
```
e supponendo che il file *dati.txt* esista e contenga la sequenza di caratteri `2 prova` (c'è uno spazio tra il carattere 2 e la stringa *prova*) seguiti dal carattere *newline*:
   a) Il programma stampa
      `2 prova`
   b) Il programma memorizza il valore 2 nella variabile *i*, poi si blocca indefinitamente sull'istruzione *f>>s ;* perché manca il carattere *newline* tra il carattere 2 e la stringa *prova* all'interno del file *dati.txt*
   c) L'*ifstream f* va in stato di errore sull'istruzione *f>>i ;*
   d) Nessuna delle altre risposte è corretta

2. **(3, -0.5)** Supponendo che la matrice `int a[5][3]` sia memorizzata a partire dall'indirizzo 100 e che gli oggetti di tipo `int` occupino 4 byte, l'indirizzo dell'elemento `a[1][0]` è
   a) 100 + 1*4*3 + 0*4 = 112
   b) 100 + 0*4*3 + 0*4 = 100
   c) 100 + 1*4*5 + 0*4 = 120
   d) nessuna delle altre risposte è corretta

3. **(3, -0.5)** Dato il seguente programma:

```
int *a, num_elem ;
void distruggi() { delete [] a; num_elem = 0 ; }
void inserisci(int n) {a[num_elem] = n ; num_elem++ ;}
void stampa() { for (int i = 0 ; i < num_elem ; i++) cout<<a[i]<<" " ; }
main(){
    a = new int[10] ;
    inserisci(1) ; stampa() ;
    distruggi() ;
    a = new int[5] ;
    inserisci(4) ; stampa() ;
}
```

   a) il programma stampa **1 4**
   b) nessuna delle altre risposte è vera
   c) il programma potrebbe essere terminato forzatamente all'atto della seconda allocazione dinamica della memoria, a causa di un errore di gestione della memoria
   d) il programma contiene un errore di gestione della memoria diverso dal *memory leak*

PARTE 2 – (POSSIBILI) RISPOSTE MULTIPLE - Ogni domanda può avere <u>da una a quattro</u> risposte CORRETTE.

- **Ogni risposta esatta viene calcolata: +1**
- **Ogni risposta errata viene calcolata: -0.5**
- **Una risposta lasciata in bianco viene calcolata: 0**

4. Durante l'esecuzione del programma in linguaggio macchina ottenuto dalla compilazione di un programma scritto in linguaggio C/C++
   a) Se in un dato istante non vi è più memoria sufficiente per creare un nuovo record di attivazione viene liberata memoria dinamica per fargli posto
   b) In ogni istante le dimensioni dello *stack* sono proporzionali al numero di funzioni invocate e non ancora terminate
   c) La scrittura al di fuori di un *array* locale ad una funzione non comporta l'aumento delle dimensioni del record di attivazione della funzione
   d) Il valore iniziale di una variabile non inizializzata e locale ad una funzione diversa dal *main* dipende solo dai valori memorizzati nella sequenza di celle di memoria riservate al programma al suo avvio

5. Quali delle seguenti affermazioni sono vere?
   a) In C non esiste il passaggio di un oggetto per riferimento, ma si può emularlo passando l'indirizzo dell'oggetto stesso
   b) La funzione **printf** della libreria standard del C determina automaticamente il formato con cui stampare ciascuno oggetto passato come argomento (dopo la stringa *format*) in base al tipo dell'oggetto stesso
   c) In C, per scrivere in una variabile un valore letto da *stdin* usando la funzione **scanf**, è necessario passare alla funzione sia l'indirizzo che il tipo della variabile in cui scrivere il valore letto
   d) Nella funzione **scanf** della libreria standard del C, sbagliare il tipo o l'indirizzo della variabile in cui si desidera scrivere il valore letto da *stdin* può causare corruzione della memoria o terminazione forzata del programma

6. Dato il seguente programma:

```
1: const int M = 10 ;
2: void fun(char a[], int n)
3: {
4:     for (int i = n - 1 ; i < M ; i++)
5:      a[i] = ' ' ;
6: }
7: main()
8: {
9:     char b[M] = "paolo" ;
10:         fun(b, 3) ;
11:         cout<<b ;
12:  }
```

a) il programma può non stampare solo **pa**
b) l'esecuzione della funzione **fun**, quando invocata alla riga 10, causa un accesso fuori dall'array, perché **M** = 10, ma la lunghezza della stringa memorizzata nell'array **b** è solo 5
c) l'istruzione alla riga 11 può stampare un numero di caratteri maggiore di 9
d) alla riga 11 la variabile **b** può non contenere più una stringa correttamente rappresentata

7. Dato il seguente programma:

```
const int N = 1000 ;
struct ss { int b; char a[N] ; } ;
void fun(ss d) { d.a[N/2] = 1 ; d.b = 3 ; cout<<d.b ; }
main()
{
 ss c ; c.b = 13 ; c.a[N/2] = 2 ;
 fun(c) ; cout<<" "<<c.b<<" "<<static_cast<int>(c.a[N/2]) ;
}
```

a) Quando **fun** è invocata il contenuto del parametro attuale **c** è copiato nel parametro formale **d**
b) In quanto al campo **a** del parametro attuale **c** passato alla funzione **fun**, l'array in esso contenuto è copiato nel campo **a** del parametro formale **d**.
c) Un passaggio per riferimento sarebbe stato più conveniente in termini di tempo di esecuzione
d) Il programma stampa: **3 13 2**

PARTE 3 – DOMANDE APERTE –

- **Una risposta esatta fa acquisire il punteggio positivo riportato a fianco della domanda**
- **Una risposta errata può eventualmente causare una penalità che dipende dalla gravità dell'errore**
- **Una risposta lasciata in bianco viene calcolata: 0**

8. **(5 pt)** Descrivere il tipo di dato strutturato in non più di sei righe.

9. **(6 pt)** Scrivere una funzione che prende in ingresso un vettore di interi e lo modifica trasformandolo in un nuovo vettore che contiene gli stessi valori interi ordinati in senso crescente, ma con eventuali valori duplicati eliminati. Ad esempio, se la funzione prende in ingresso il vettore [10, 4, 6, 3, 4, 1, 10, 6], allora trasforma il vettore in [1, 3, 4, 6, 10].

Programmazione I
Prova scritta - 18 settembre 2013

Nome: _____ Cognome: _____

Matricola: _____

Indicare le risposte corrette apponendo una croce nella casella corrispondente. Per superare la prova bisogna aver raggiunto almeno 9 punti nelle domande a risposta singola/multipla, ed almeno 15 complessivamente. Questa è l'unica pagina che dovete consegnare. Per comodità avete anche un copia di questa pagina per calcolare il voto da sole/soli durante la correzione.

	Risposte				**Punti/**	
	A	**B**	**C**	**D**	**Penalità**	
1					**3**	**-0,5**
2					**3**	**-0,5**
3					**3**	**-0,5**
4						
5						
6						
7						

Risposta alla domanda 8 (5 pt):

Risposta alla domanda 9 (6 pt):

Programmazione I
Prova scritta - 18 settembre 2013

Nome: _____ Cognome: _____

Matricola: _____

Indicare le risposte corrette apponendo una croce nella casella corrispondente. Per superare la prova bisogna aver raggiunto almeno 9 punti nelle domande a risposta singola/multipla, ed almeno 15 complessivamente. Usate questa copia per calcolare il voto da sole/soli durante la correzione.

	Risposte				Punti/ Penalità	
	A	**B**	**C**	**D**		
1					**3**	**-0,5**
2					**3**	**-0,5**
3					**3**	**-0,5**
4						
5						
6						
7						

Risposta alla domanda 8 (5 pt):

Risposta alla domanda 9 (6 pt):