Esame di Programmazione I e Informatica Generale I

05 febbraio 2003
Prof. Sebastiani

Codice: .....

Nome	Cognome	Matricola

La directory 'esame' contiene 5 sotto-directory: 'uno', 'due', 'tre', 'quattro' e 'cinque'. Le soluzioni vanno scritte negli spazie e nei modi indicati esercizio per esercizio.
NOTA: il codice dato non può essere modicicato

Modalità d'Esame

Durante la prova gli studenti sono vincolati a seguire le regole seguenti:

- Non è consentito l'uso di alcun libro di testo o fotocopia. In caso lo studente necessitasse di carta (?), gli/le verranno forniti fogli di carta bianca su richiesta, che dovranno essere riconsegnati a fine prova. È consentito l'uso di una penna. Non è consentito l'uso di alcuno strumento calcolatore.
- È vietato lo scambio di qualsiasi informazione, orale o scritta. È vietato guardare nel terminale del vicino.
- È vietato l'uso di telefoni cellulari o di qualsiasi strumento elettronico.
- È vietato allontanarsi dall'aula durante la prova, anche se si ha già consegnato. (Ogni necessità fisiologica va espletata PRIMA dell'inizio della prova.)
- È vietato qualunque accesso, in lettura o scrittura, a file esterni alla directory di lavoro assegnata a ciascun studente. Le uniche operazioni consentite sono l'apertura, l'editing, la copia, la rimozione e la compilazione di file all'interno della propria directory di lavoro.
- Sono ovviamente vietati l'uso di email, ftp, ssh, telnet ed ogni strumento che consenta di accedere a file esterni alla directory di lavoro. Le operazioni di copia, rimozione e spostamento di file devono essere circoscritte alla directory di lavoro.
- Ogni altra attività non espressamente citata qui sopra o autorizzata dal docente è vietata.

Ogni violazione delle regole di cui sopra comporterà automaticamente l'annullamento della prova e il divieto di accesso ad un certo numero di appelli successivi, a seconda della gravità e della recidività della violazione.

NOTA IMPORTANTE: DURANTE LA PROVA PER OGNI STUDENTE VERRÀ ATTIVATO UN TRACCIATORE SOFTWARE CHE REGISTRERÀ TUTTE LE OPERAZIONI ESEGUITE (ANCHE ALL'INTERNO DELL'EDITOR!!). L'ANNULLAMENTO DELLA PROVA DI UNO STUDENTE POTRÀ AVVENIRE ANCHE IN UN SECONDO MOMENTO, SE L'ANALISI DELLE TRACCE SOFTWARE RIVELASSERO IRREGOLARITÀ.

1 Dato il seguente main:

```
//definizione della funzione f

void main ()
{
  int x;

  cout << "Inserisci un numero: ";
  cin >> x;

  cout << f(x) << endl;
}
```

scrivere la definizione della funzione `f`:

```
int f (int x);
```

tale che, preso in input un valore intero `x`, restituisca i seguenti valori calcolati in x:

$$\begin{cases} 1 & \text{se } x \text{ è pari} \\ 0 & \text{se } x \text{ è dispari} \\ -1 & \text{se } x \text{ è } 0 \end{cases}$$

1 Dato il seguente main:

```
//definizione della funzione f

void main ()
{
  int x;

  cout << "Inserisci un numero: ";
  cin >> x;

  cout << f(x) << endl;
}
```

scrivere la definizione della funzione `f`:

```
int f (int x);
```

tale che, preso in input un valore intero `x`, restituisca i seguenti valori calcolati in x:

$$\begin{cases} 1 & \text{se } x \text{ è dispari} \\ 0 & \text{se } x \text{ è } 0 \\ -1 & \text{se } x \text{ è pari} \end{cases}$$

1 Dato il seguente main:

```
//definizione della funzione f

void main ()
{
  int x;

  cout << "Inserisci un numero: ";
  cin >> x;

  cout << f(x) << endl;
}
```

scrivere la definizione della funzione `f`:

```
int f (int x);
```

tale che, preso in input un valore intero `x`, restituisca i seguenti valori calcolati in x:

$$\begin{cases} 1 & \text{se } x \text{ è dispari} \\ 2 & \text{se } x \text{ è pari} \\ 0 & \text{se } x \text{ è } 0 \end{cases}$$

1 Dato il seguente main:

```
//definizione della funzione f

void main ()
{
  int x;

  cout << "Inserisci un numero: ";
  cin >> x;

  cout << f(x) << endl;
}
```

scrivere la definizione della funzione `f`:

```
int f (int x);
```

tale che, preso in input un valore intero `x`, restituisca i seguenti valori calcolati in x:

$$\begin{cases} 1 & \text{se } x \text{ è pari} \\ 0 & \text{se } x \text{ è } 0 \\ -1 & \text{se } x \text{ è dispari} \end{cases}$$

2 Dato il seguente main:

```
//definizione della funzione quanto_divisibile

void main ()
{
  //Dichiarazione variabili
  int x,risultato;

  //Richiesta e acquisizione dati
  cout << "Inserisci un valore intero: \n";
  cin >> x;

  //Calcolo di quante volte un valore intero
  //e' divisibile per due (chiamata di funzione)
  risultato = quanto_divisibile(x);

  //Stampa dei risultati
  cout << "Il valore "<<x<< " e' divisibile per due "<<risultato<< " volte."<<endl;
}
```

scrivere la definizione della funzione `quanto_divisibile`:

```
int quanto_divisibile (int x);
```

in grado di calcolare e ritornare quante volte un intero `x` è divisibile per due.

2 Dato il seguente main:

```
//definizione della funzione quanto_divisibile

void main ()
{
  //Dichiarazione variabili
  int x,risultato;

  //Richiesta e acquisizione dati
  cout << "Inserisci un valore intero: \n";
  cin >> x;

  //Calcolo di quante volte un valore intero
  //e' divisibile per tre (chiamata di funzione)
  risultato = quanto_divisibile(x);

  //Stampa dei risultati
  cout << "Il valore "<<x<< " e' divisibile per tre "<<risultato<< " volte."<<endl;
}
```

scrivere la definizione della funzione `quanto_divisibile`:

```
int quanto_divisibile (int x);
```

in grado di calcolare e ritornare quante volte un intero `x` è divisibile per tre.

2 Dato il seguente main:

```
//definizione della funzione quanto_divisibile

void main ()
{
  //Dichiarazione variabili
  int x,risultato;

  //Richiesta e acquisizione dati
  cout << "Inserisci un valore intero: \n";
  cin >> x;

  //Calcolo di quante volte un valore intero
  //e' divisibile per cinque (chiamata di funzione)
  risultato = quanto_divisibile(x);

  //Stampa dei risultati
  cout << "Il valore "<<x<< " e' divisibile per cinque "<<risultato<< " volte."<<endl;
}
```

scrivere la definizione della funzione `quanto_divisibile`:

```
int quanto_divisibile (int x);
```

in grado di calcolare e ritornare quante volte un intero `x` è divisibile per cinque.

2 Dato il seguente main:

```
//definizione della funzione quanto_divisibile

void main ()
{
  //Dichiarazione variabili
  int x,risultato;

  //Richiesta e acquisizione dati
  cout << "Inserisci un valore intero: \n";
  cin >> x;

  //Calcolo di quante volte un valore intero
  //e' divisibile per sette (chiamata di funzione)
  risultato = quanto_divisibile(x);

  //Stampa dei risultati
  cout << "Il valore "<<x<< " e' divisibile per sette "<<risultato<< " volte."<<endl;
}
```

scrivere la definizione della funzione `quanto_divisibile`:

```
int quanto_divisibile (int x);
```

in grado di calcolare e ritornare quante volte un intero `x` è divisibile per sette.

3 Dato il seguente main:

```
//definizione delle funzioni leggi_array e cerca_max

int main()
{
  double vettore[100], max;
  int dim, indice_del_max;

  cout << "Inserisci la dimensione del vettore: " << endl;
  cin >> dim;

  leggi_array (vettore,dim);

  indice_del_max = cerca_max (vettore, dim, max);

  cout << "In base ai dati inseriti nel vettore il valore"<<endl;
  cout << "del seno e' massimo con il valore "<<max<<endl;
  cout << "tale valore ha indice "<<indice_del_max<<endl;
}
```

e la funzione `leggi_array` per l'inserimento in un vettore di un dato numero di reali:

```
void leggi_array (double vettore[], int dim);
```

scrivere la definizione della funzione `cerca_max` che, presi in input l'array di reali `vettore`, la dimensione `dim` e una variabile `max`, sia in grado di calcolare per quale valore memorizzato nell'array si ottinene il seno massimo. Tale funzione dovrà memorizzare nella variabile `max` questo valore letto dall'array di reali per cui il seno è massimo e dovrà restituire la posizione di detto valore nell'array.

Si utilizzi la seguente funzione della libreria `math.h`:

```
double sin (double x);
```

che restituisce rispettivamente il seno di un numero reale `x` passato come parametro, dove `x` è espresso in radianti.

3 Dato il seguente main:

```
//definizione delle funzioni leggi_array e cerca_min

int main()
{
  double vettore[100], min;
  int dim, indice_del_min;

  cout << "Inserisci la dimensione del vettore: " << endl;
  cin >> dim;

  leggi_array (vettore,dim);

  indice_del_min = cerca_min (vettore, dim, min);

  cout << "In base ai dati inseriti nel vettore il valore"<<endl;
  cout << "del seno e' minimo con il valore "<<min<<endl;
  cout << "tale valore ha indice "<<indice_del_min<<endl;
}
```

e la funzione `leggi_array` per l'inserimento in un vettore di un dato numero di reali:

```
void leggi_array (double vettore[], int dim);
```

scrivere la definizione della funzione `cerca_min` che, presi in input l'array di reali `vettore`, la dimensione `dim` e una variabile `min`, sia in grado di calcolare per quale valore memorizzato nell'array si ottinene il seno minimo. Tale funzione dovrà memorizzare nella variabile `min` questo valore letto dall'array di reali per cui il seno è minimo e dovrà restituire la posizione di detto valore nell'array.

Si utilizzi la seguente funzione della libreria `math.h`:

```
double sin (double x);
```

che restituisce rispettivamente il seno di un numero reale `x` passato come parametro, dove `x` è espresso in radianti.

3 Dato il seguente main:

```
//definizione delle funzioni leggi_array e cerca_max

int main()
{
  double vettore[100], max;
  int dim, indice_del_max;

  cout << "Inserisci la dimensione del vettore: " << endl;
  cin >> dim;

  leggi_array (vettore,dim);

  indice_del_max = cerca_max (vettore, dim, max);

  cout << "In base ai dati inseriti nel vettore il valore"<<endl;
  cout << "del coseno e' massimo con il valore "<<max<<endl;
  cout << "tale valore ha indice "<<indice_del_max<<endl;
}
```

e la funzione `leggi_array` per l'inserimento in un vettore di un dato numero di reali:

```
void leggi_array (double vettore[], int dim);
```

scrivere la definizione della funzione `cerca_max` che, presi in input l'array di reali `vettore`, la dimensione `dim` e una variabile `max`, sia in grado di calcolare per quale valore memorizzato nell'array si ottinene il coseno massimo. Tale funzione dovrà memorizzare nella variabile `max` questo valore letto dall'array di reali per cui il coseno è massimo e dovrà restituire la posizione di detto valore nell'array.

Si utilizzi la seguente funzione della libreria `math.h`:

```
double cos (double x);
```

che restituisce rispettivamente il coseno di un numero reale `x` passato come parametro, dove `x` è espresso in radianti.

3 Dato il seguente main:

```
//definizione delle funzioni leggi_array e cerca_min

int main()
{
  double vettore[100], min;
  int dim, indice_del_min;

  cout << "Inserisci la dimensione del vettore: " << endl;
  cin >> dim;

  leggi_array (vettore,dim);

  indice_del_min = cerca_min (vettore, dim, min);

  cout << "In base ai dati inseriti nel vettore il valore"<<endl;
  cout << "del coseno e' minimo con il valore "<<min<<endl;
  cout << "tale valore ha indice "<<indice_del_min<<endl;
}
```

e la funzione `leggi_array` per l'inserimento in un vettore di un dato numero di reali:

```
void leggi_array (double vettore[], int dim);
```

scrivere la definizione della funzione `cerca_min` che, presi in input l'array di reali `vettore`, la dimensione `dim` e una variabile `min`, sia in grado di calcolare per quale valore memorizzato nell'array si ottinene il coseno minimo. Tale funzione dovrà memorizzare nella variabile `min` questo valore letto dall'array di reali per cui il coseno è minimo e dovrà restituire la posizione di detto valore nell'array.

Si utilizzi la seguente funzione della libreria `math.h`:

```
double cos (double x);
```

che restituisce rispettivamente il coseno di un numero reale `x` passato come parametro, dove `x` è espresso in radianti.

4 Dato il seguente main:

```
//definizione delle funzioni leggi_matrice, and_matrice e stampa_matrice

void main()
{
  int matrice1[100][100], matrice2[100][100];
  int righe1,colonne1,righe2,colonne2,errore;

  cout << "Quante righe ha la prima matrice? ";
  cin >> righe1;
  cout << "Quante colonne ha la prima matrice? ";
  cin >> colonne1;

  leggi_matrice (matrice1,righe1,colonne1);

  cout << "Quante righe ha la seconda matrice? ";
  cin >> righe2;
  cout << "Quante colonne ha la seconda matrice? ";
  cin >> colonne2;

  leggi_matrice (matrice2,righe2,colonne2);

  errore = and_matrice (matrice1,righe1,colonne1,matrice2,righe2,colonne2);

  if (errore==-1)
    cout<<"Errore: le matrici inserite non hanno le stesse dimensioni!!"<<endl;
  else
    {
      cout<<"La matrice risultante e':"<<endl;
      stampa_matrice(matrice1,righe1,colonne1);
    }
}
```

scrivere la definizione della funzione:

```
int and_matrice (int matrice1[][100],int righe1,int colonne1,
                 int matrice2[][100],int righe2,int colonne2);
```

che, prese in input due matrici di booleani (0 e 1) `matrice1` e `matrice2` con i relativi numeri di colonne e righe, calcola la matrice "AND" di `matrice1` e `matrice2` e la memorizza in `matrice1`. La matrice "AND" viene calcolata in base alla tabella di verità riporata qui sotto. In particolare alla posizione `[i][j]` di `matrice1` dovrà essere memorizzato il valore dettato dalla tabella di verità considerando come operatori gli elementi `[i][j]` di `matrice1` e di `matrice2`. Una volta eseguita questa operazione per tutti gli elementi di `matrice1` la funzione dovrà restiutire 1. Nel caso di matrici con dimensioni diverse la funzione dovrà restituire -1.

Tabella di Verità:

Operatore 1	Operatore 2	Risultato
0	0	0
0	1	0
1	0	0
1	1	1

4 Dato il seguente main:

```
//definizione delle funzioni leggi_matrice, or_matrice e stampa_matrice

void main()
{
  int matrice1[100][100], matrice2[100][100];
  int righe1,colonne1,righe2,colonne2,errore;

  cout << "Quante righe ha la prima matrice? ";
  cin >> righe1;
  cout << "Quante colonne ha la prima matrice? ";
  cin >> colonne1;

  leggi_matrice (matrice1,righe1,colonne1);

  cout << "Quante righe ha la seconda matrice? ";
  cin >> righe2;
  cout << "Quante colonne ha la seconda matrice? ";
  cin >> colonne2;

  leggi_matrice (matrice2,righe2,colonne2);

  errore = or_matrice (matrice1,righe1,colonne1,matrice2,righe2,colonne2);

  if (errore==-1)
    cout<<"Errore: le matrici inserite non hanno le stesse dimensioni!!"<<endl;
  else
    {
      cout<<"La matrice risultante :"<<endl;
      stampa_matrice(matrice1,righe1,colonne1);
    }
}
```

scrivere la definizione della funzione:

```
int or_matrice (int matrice1[][100],int righe1,int colonne1,
                int matrice2[][100],int righe2,int colonne2);
```

che, prese in input due matrici di booleani (0 e 1) `matrice1` e `matrice2` con i relativi numeri di colonne e righe, calcola la matrice "OR" di `matrice1` e `matrice2` e la memorizza in `matrice1`. La matrice "OR" viene calcolata in base alla tabella di verità riporata qui sotto. In particolare alla posizione `[i][j]` di `matrice1` dovrà essere memorizzato il valore dettato dalla tabella di verità considerando come operatori gli elementi `[i][j]` di `matrice1` e di `matrice2`. Una volta eseguita questa operazione per tutti gli elementi di `matrice1` la funzione dovrà restiutire 1. Nel caso di matrici con dimensioni diverse la funzione dovrà restituire -1.

Tabella di Verità:

Operatore 1	Operatore 2	Risultato
0	0	0
0	1	1
1	0	1
1	1	1

4 Dato il seguente main:

```
//definizione delle funzioni leggi_matrice, nand_matrice e stampa_matrice

void main()
{
  int matrice1[100][100], matrice2[100][100];
  int righe1,colonne1,righe2,colonne2,errore;

  cout << "Quante righe ha la prima matrice? ";
  cin >> righe1;
  cout << "Quante colonne ha la prima matrice? ";
  cin >> colonne1;

  leggi_matrice (matrice1,righe1,colonne1);

  cout << "Quante righe ha la seconda matrice? ";
  cin >> righe2;
  cout << "Quante colonne ha la seconda matrice? ";
  cin >> colonne2;

  leggi_matrice (matrice2,righe2,colonne2);

  errore = nand_matrice (matrice1,righe1,colonne1,matrice2,righe2,colonne2);

  if (errore==-1)
    cout<<"Errore: le matrici inserite non hanno le stesse dimensioni!!"<<endl;
  else
    {
      cout<<"La matrice risultante :"<<endl;
      stampa_matrice(matrice1,righe1,colonne1);
    }
}
```

scrivere la definizione della funzione:

```
int nand_matrice (int matrice1[][100],int righe1,int colonne1,
                  int matrice2[][100],int righe2,int colonne2);
```

che, prese in input due matrici di booleani (0 e 1) `matrice1` e `matrice2` con i relativi numeri di colonne e righe, calcola la matrice "NAND" di `matrice1` e `matrice2` e la memorizza in `matrice1`. La matrice "NAND" viene calcolata in base alla tabella di verità riporata qui sotto. In particolare alla posizione `[i][j]` di `matrice1` dovrà essere memorizzato il valore dettato dalla tabella di verità considerando come operatori gli elementi `[i][j]` di `matrice1` e di `matrice2`. Una volta eseguita questa operazione per tutti gli elementi di `matrice1` la funzione dovrà restiutire 1. Nel caso di matrici con dimensioni diverse la funzione dovrà restituire -1.

Tabella di Verità:

Operatore 1	Operatore 2	Risultato
0	0	1
0	1	1
1	0	1
1	1	0

4 Dato il seguente main:

```
//definizione delle funzioni leggi_matrice, nor_matrice e stampa_matrice

void main()
{
  int matrice1[100][100], matrice2[100][100];
  int righe1,colonne1,righe2,colonne2,errore;

  cout << "Quante righe ha la prima matrice? ";
  cin >> righe1;
  cout << "Quante colonne ha la prima matrice? ";
  cin >> colonne1;

  leggi_matrice (matrice1,righe1,colonne1);

  cout << "Quante righe ha la seconda matrice? ";
  cin >> righe2;
  cout << "Quante colonne ha la seconda matrice? ";
  cin >> colonne2;

  leggi_matrice (matrice2,righe2,colonne2);

  errore = nor_matrice (matrice1,righe1,colonne1,matrice2,righe2,colonne2);

  if (errore==-1)
    cout<<"Errore: le matrici inserite non hanno le stesse dimensioni!!"<<endl;
  else
    {
      cout<<"La matrice risultante :"<<endl;
      stampa_matrice(matrice1,righe1,colonne1);
    }
}
```

scrivere la definizione della funzione:

```
int nor_matrice (int matrice1[][100],int righe1,int colonne1,
                 int matrice2[][100],int righe2,int colonne2);
```

che, prese in input due matrici di booleani (0 e 1) `matrice1` e `matrice2` con i relativi numeri di colonne e righe, calcola la matrice "NOR" di `matrice1` e `matrice2` e la memorizza in `matrice1`. La matrice "NOR" viene calcolata in base alla tabella di verità riporata qui sotto. In particolare alla posizione `[i][j]` di `matrice1` dovrà essere memorizzato il valore dettato dalla tabella di verità considerando come operatori gli elementi `[i][j]` di `matrice1` e di `matrice2`. Una volta eseguita questa operazione per tutti gli elementi di `matrice1` la funzione dovrà restiutire 1. Nel caso di matrici con dimensioni diverse la funzione dovrà restituire -1.

Tabella di Verità:

Operatore 1	Operatore 2	Risultato
0	0	1
0	1	0
1	0	0
1	1	0

5 ESERCIZIO FACOLTATIVO:
Preparare un programma che consenta di giocare a Master Mind con le seguenti regole:

- il computer genera un codice segreto formato da una serie di quattro colori scelti tra i sei colori (indaco, arancio, giallo, verde, cobalto e rosso);
- il giocatore cerca di individuare il codice attraverso una serie di tentativi;
- in risposta a ciascun tentativo il computer scrive "nero" per ogni colore giusto al posto giusto e scrive "bianco" per ogni colore presente ma al posto sbagliato;
- ogni giocatore ha sei tentativi a disposizione per individuare la sequenza di colori altrimenti ha perso.

Nello sviluppo del codice utilizzare per l'input e l'output dei dati le funzioni qui sotto riportate.

```
void inizializza (void)
{
  char nome,cognome;
  int seme;

  cout<<"Inserisci le iniziali del giocatore: "<<endl;;
  cin>>nome;
  cin>>cognome;
  seme=(int)(nome)+(int)(cognome);
  srand(seme);
}
void inserisci_codice (char codice_scelto[])
{
  cout<<"Inserisci una possibile combinazione:"<<endl;
  for (int i=0;i<4;i++)
    cin>>codice_scelto[i];
}
void stampa_risultati(int numero_neri, int numero_bianchi, int numero_tentativo)
{
  cout<<"Neri: "<<numero_neri<<" Bianchi: "<<numero_bianchi<<endl;
  if (numero_neri==4)
    cout<<"HAI VINTO!!!"<<endl;
  if (numero_tentativo==6)
    cout<<"HAI PERSO!!!"<<endl;
}
```

Supponendo che il computer abbia generato il seguente codice da indovinare: A G I V sono di seguito riportate due sessioni di esempio nel caso di vittoria e di sconfitta:

```
Inserire una possibile combinazione:
R G C C
Neri: 1 Bianchi: 0
Inserire una possibile combinazione:
A G C I
Neri: 2 Bianchi: 1
Inserire una possibile combinazione:
A G I V
Neri: 4 Bianchi: 0
HAI VINTO!!!


Inserire una possibile combinazione:
C C R R
Neri: 0 Bianchi: 0
Inserire una possibile combinazione:
V V V V
Neri: 1 Bianchi: 0
Inserire una possibile combinazione:
I I I V
Neri: 2 Bianchi: 0
Inserire una possibile combinazione:
G I I V
Neri: 2 Bianchi: 1
Inserire una possibile combinazione:
I I G V
Neri: 1 Bianchi: 2
Inserire una possibile combinazione:
I G I V
Neri: 3 Bianchi: 0
HAI PERSO!!!
```