POLITECNICO
MILANO 1863

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA

CORSO DI LAUREA IN INGEGNERIA AEROSPAZIALE

Prof. Gerardo Pelosi

Informatica – a.a. 2020/2021 – 1° Appello – 25 Giugno 2021

Cognome _____ **Matricola o Cod. Persona** _____

Nome _____ **Firma** _____

Istruzioni

- Non separate questi fogli. Scrivete la soluzione **solo sui fogli distribuiti**, utilizzando il retro delle pagine se necessario. **Cancellate le parti di brutta** (o ripudiate) con un tratto di **penna**.
- Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.
- **È possibile scrivere a matita** e non occorre ricalcare al momento della consegna.
- È **vietato** utilizzare **calcolatrici** e qualsiasi **dispositivo elettronico**. Chi tenti di farlo vedrà **annullata** la sua prova.
- È **vietato** consultare **libri o appunti**. Chi tenti di farlo vedrà **annullata** la sua prova.
- Qualsiasi **tentativo** di comunicare con altri studenti comporta **l'espulsione** dall'aula.
- Non è possibile lasciare l'aula conservando il tema della prova in corso.
- È possibile ritirarsi senza penalità.
- Tempo a disposizione: **2h:30min**

Valore indicativo degli esercizi, voti parziali e voto finale:

Esercizio 1 (6 punti) _____

Esercizio 2 (10 punti) _____

Esercizio 3 (8 punti) _____

Esercizio 4 (8 punti) _____

Totale (32 punti) _____

Voto finale _____

Esercizio 1 Codifiche numeriche e algebra di Boole [6 punti]

1. Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i valori numerici seguenti: A = **32_{hex}** (numero positivo in notazione esadecimale) e B = **80_{dec}** (numero positivo in notazione decimale).
   (a) Si esibisca la conversione di A e B in *complemento a due*.
   (b) Si esibisca il risultato della somma (B+A) e la differenza (B–A) svolte considerando gli operandi codificati in in *complemento a due*, e si indichi se nell'esecuzione di ciascuna operazione si genera un riporto oltre la posizione del bit di segno e se si verifica *overflow*.
   [4 punti]

2. Si riscriva l'espressione condizionale del seguente costrutto iterativo in linguaggio C facente uso di tre variabili intere a, b, c:

```
while( (a < 7 && b < 12) || (b < 30 && c < 0)) {
      ...
}
```

   in modo tale che la condizione risultante risulti equivalente alla prima e non faccia uso dell'operatore `'<'`. **[2 punti]**

Soluzione

1. La soluzione richiesta prevede quanto segue.

   $A = 32_{hex} = 50_{dec}$ (cinquanta) $= 0011\ 0010_{bin} = 110010_{bin}$
   $B = 80_{dec}$ = ottanta = 1010000_{bin}

   7 bit sono strettamente necessari per rappresentare entrambi i valori in binario naturale.
   Il numero minimo di bit per rappresentare entrambi in *complemento a due* è dunque 8 bit.

   $B = 1010000_{bin} = 01010000_{c2}$
   $A = 110010_{bin} = 00110010_{c2}$
   $B+A = 10000010_{c2}$ (NO riporto oltre la posizione del bit di segno;
   addendi di segno concorde e risultato con segno discorde → *Overflow*)

   B-A = B + (-A)

   $B = 1010000_{bin} = 01010000_{c2}$
   $-A = not(00110010_{c2}) + 1 = 11001110_{c2}$
   $B-A = 00011110_{c2}$ (SI riporto oltre la posizione del bit di segno;
   addendi di segno discorde → NO *Overflow*)

2. L'espressione condizionale

```
(    (a < 7 && b < 12) || (b < 30 && c < 0)    )
```

   si può esprimere in maniera equivalente negando due volte l'intera espressione e applicando le leggi di De Morgan):

```
!(   !(a < 7 && b < 12) && !(b < 30 && c < 0)   )
```

   ottenendo

```
!( (a >= 7 || b >= 12) && (b >= 30 || c >=0) )
```

Esercizio 2 [10 punti]

Si consideri un numero intero positivo `N` e il valore della somma dei suoi divisori `S`, escludendo da tale somma il numero stesso (perché un qualunque numero è sì un divisore di se stesso, ma improprio).
Chiamiamo ***difettivo*** un numero intero tale che la somma dei suoi divisori è minore del numero stesso (per esempio: `N=10` è difettivo perché `S=1+2+5=8`).
Chiamiamo ***perfetto*** un numero intero uguale alla somma dei suoi divisori (per esempio: `N=6` è perfetto perché `S=1+2+3=6`).
Chiamiamo ***abbondante*** un numero intero tale che la somma dei suoi divisori è maggiore del numero stesso (per esempio: `N=42` è abbondante perché `S=1+2+3+6+7+14+21=54`).

(a) Si codifichino in `C` le funzioni `...somma_divisori(...)` e `...classifica_numero(...)`, ciascuna delle quali riceve come parametro un numero intero positivo.
La prima funzione restituisce la somma dei divisori del valore ricevuto come argomento (non considerando il numero stesso come divisore).
La seconda funzione restituisce `-1` se il valore ricevuto come argomento è difettivo, restituisce `0` se è perfetto, oppure restituisce `+1` se è abbondante. **[6 punti]**

Nota: si assuma che sia responsabilità del programma chiamante garantire che il valore passato come parametro debba essere strettamente positivo (quindi né negativo, né nullo).

Soluzione 2(a)

```
int somma_divisori(int x) {

     int somma = 1;
     for (int i = 2; i <= x/2; i++) {
         if (x % i == 0)
           somma += i;
     }
} // end somma_divisori



int classifica_numero(int x) {

     int s = somma_divisori(x);

     if (s < x) return -1;
     if (s == x) return 0;
     return +1;
} // end classifica_numero
```

(a) Si codifichi in C un programma principale, **int** `main(){...}`, che acquisisca dallo *standard input* una sequenza di numeri interi (di lunghezza arbitraria) terminata dal valore `0` e che riporti sullo *standard output* il conteggio delle coppie di numeri consecutivi ed entrambi difettivi, oppure consecutivi ed entrambi perfetti oppure consecutivi ed entrambi abbondandi. **[4 punti]**

Nota: quando appropriato, è ammesso il riutilizzo dei sottoprogrammi definiti al punto (a).

Soluzione 2(b)

```
#include <stdio.h>

int main(){

   int difettivi = 0, perfetti = 0, abbondanti = 0;
   int prev, curr;

   printf("\n Inserisci una sequenza di interi separati da spazio ");
   printf(" e terminata da 0:\n");

   scanf("%d", &curr);
   while (curr != 0) {
     prev = curr;
     scanf("%d", &curr);

     if (prev > 0 && curr > 0) {
       int p = classifica_numero(prev);
       int c = classifica_numero(curr);
       if (p == c) {
         if (p == -1) difettivi += 1;
         else if (p == 0) perfetti += 1;
              else abbondanti += 1;
       }
     }
   } // end while

   printf("\n Num. di valori consecutivi e difettivi: %d", difettivi);
   printf("\n Num. di valori consecutivi e perfetti: %d", perfetti);
   printf("\n Num. di valori consecutivi e abbondanti: %d", abbondanti);
   printf("\n Num. di valori validi: %d", difettivi+perfetti+abbondanti);

} // end main
```

Esercizio 3 [8 punti]

(a) Scrivere in C un sottoprogramma con il prototipo seguente

```
int equiv(char prima[], char seconda[])
```

che presi come argomenti due stringhe, restituisca +1 se la seconda stringa risulta essere uguale alla prima dopo aver effettuato uno scambio della prima lettera della seconda stringa con un'altra lettera della stessa.

Esempio:

```
prima:     "ANNA", seconda: "NANA"      → restituisce +1
prima:     "MARE", seconda: "RAME"      → restituisce +1
prima:   "ERCOLE", seconda: "CREOLE"    → restituisce +1
prima:     "ANNA", seconda: "ANNA"      → restituisce +1
prima:"BALLETTO", seconda: "BOLLETTA"   → restituisce 0
prima:     "ARCO", seconda: "ARCA"      → restituisce 0
```

(b) Scrivere in C un sottoprogramma con il prototipo seguente

```
int equivalenti(char prima[], char seconda[])
```

che presi come argomenti due stringhe, restituisca +1 se risultano essere uguali effettuando al più uno scambio di due caratteri (in posizioni qualsiasi), 0 altrimenti.

Esempio:

```
prima: "BALLETTO", seconda: "BOLLETTA"  → restituisce +1
prima: "CARIATO",  seconda: "CARTAIO"   → restituisce +1
prima: "SARTO",    seconda: "TARSO"     → restituisce +1
prima: "ANNO",     seconda: "ANNO"      → restituisce +1
prima: "ARCO",     seconda: "ARCA"      → restituisce 0
```

Nota: è ammeso far uso delle funzioni della libreria `<string.h>`

Se si continua sul retro di qualche foglio, indicare quale

Soluzione 3(a)

```
#include <string.h>

int equiv(char prima[], char seconda[]) {
   char tmp;

   for (int i = 1; i < strlen(seconda); i++) {
      tmp = seconda[0]; seconda[0] = seconda[i]; seconda[i] = tmp;
      if (strcmp(prima, seconda) == 0)
        return +1;
      tmp = seconda[0]; seconda[0] = seconda[i]; seconda[i] = tmp;
   }
   return 0;
} // end equiv
```

Soluzione 3(b)

```
#include <string.h>

int equivalenti(char prima[], char seconda[]) {
   char tmp;

   for (int j = 0; j < strlen(seconda)-1; j++) {
      for (int i = j+1; i < strlen(seconda); i++) {
         tmp = seconda[j]; seconda[j] = seconda[i]; seconda[i] = tmp;
         if (strcmp(prima, seconda) == 0)
           return +1;
         tmp = seconda[j]; seconda[j] = seconda[i]; seconda[i] = tmp;
      }
   }
   return 0;
} // end equivalenti
```

Esercizio 4 [8 punti]

Nello sport dei tuffi individuali, ogni tuffo viene valutato da **cinque giudici** diversi.
Ogni giudice esprime un **punteggio individuale** da 0 a 10, con la possibilità di esprimere anche mezzi punti.
Il punteggio 0 corrisponde a un tuffo completamente sbagliato; il punteggio 10 a un tuffo perfetto.
Per eliminare eventuali preferenze soggettive, il punteggio **più alto** e quello **più basso** vengono scartati in automatico.
Ogni punteggio è espresso indipendentemente dal **grado di difficoltà** del tuffo (un valore numerico che va da 1.3 a 3.6) e il punteggio finale attribuito all'atleta è calcolato come prodotto della somma dei tre punteggi rimanenti per il grado di difficoltà del tuffo.

1. Date le seguenti definizioni di costanti e di tipo di dato

```
#define NUM_GIUDICI 5
#define MAX_STRING 30

typedef struct {
                char  nome[MAX_STRING+1];
                char  cognome[MAX_STRING+1];
                char  sesso;
                char  nazione[MAX_STRING+1];
                float punteggi[NUM_GIUDICI];
                float gradoDifficolta;
        } atleta_t;

typedef struct {
                   atleta_t* array;
                   int numAtleti;
        } gara_t;
```

   scrivere in C la funzione

```
gara_t acquisisciDaFile(char nomefile[])
```

   che restituisce una variabile di tipo gara_t che memorizzi come valore del suo primo attributo l'indirizzo iniziale di un *array* dinamico con tante celle quante sono le righe del *file* di testo il cui nome è indicato come argomento del sottoprogramma, e come valore del suo secondo attributo la lunghezza dell'*array* dinamico.
   Il *file* di testo è assunto avere il seguente formato:

```
Tania Cagnotto F ITALIA  10 9.2 8.5 8.2 9.5 3.0
Kent Ferguson M USA 9.8 9.1 8.2 9.5 9.0 2.8
Dmitrij Sautin M RUSSIA 9.8 9.5 8.0 8.5 9.0 2.8
Patrick Hausding M GERMANIA 9.5 9.5 9.5 10 8.0 1.5
Shi Tingmao F CINA 8.5 8.2 8.0 9.0 9.5 2.5
…
```

   dove su ogni rigo sono riportati nome, cognome, sesso ('M' per uomini e 'F' per donne), nazione di un atleta, il punteggio a lui assegnato da ciascuno dei NUM_GIUDICI giudici e il grado di difficoltà del tuffo eseguito. Ogni cella dell'*array* dinamico deve contenere le informazioni prese da un rigo diverso del *file*.

   N.B. Si assuma che non ci siano nomi o cognomi composti da più di una parola. Le informazioni su ogni rigo sono separate da un carattere ' ' (spazio). Il numero di righi del *file* non è noto a priori.

2. Scrivere in C la funzione

```
void stampa_migliori(gara_t g)
```

   che riporti a schermo il nome di uno dei migliori atleti della competizione (cioè uno di quelli che hanno ottenuto il punteggio più alto).

Se si continua sul retro di qualche foglio, indicare quale

Soluzione 4(a)

```
#include <stdlib.h>

gara_t acquisisciDaFile(char nomefile[]) {

   FILE* fp = fopen(nomefile, "r");
   if (fp == NULL) {
     printf("\n Errore nell'apertura del file %s. ", nomefile);
     printf("\n Programma terminato!\n");
     exit(-1);
   }

   int numRighe = 0;
   atleta_t a;

   while (feof(fp) == 0) {
     fscanf(fp, "%s %s %c %s", a.nome, a.cognome, &a.genere, a.nazione);
     for (int i = 0; i < NUM_GIUDICI; i++)
        fscanf(fp, "%f", &a.punteggi[i]);
     fscanf(fp, "%f", &a. gradoDifficolta);

     numRighe += 1;
   } // end while

   if (numRighe == 0) {
     printf("\n Il file e' vuoto! ");printf("\n Programma terminato!\n");
     exit(-1);
   }

   gara_t g;
   g.array = (atleta_t*)malloc(sizeof(atleta_t)*numRighe);
   if (g.array == NULL) {
     printf("\n malloc(...) fallita! ");
     printf("\n Programma terminato!\n");
     exit(-1);
   }
   g.numAtleti = numRighe;

   rewind(fp);
   int k = 0; // indice delle celle dell'array dinamico
   while (feof(fp) == 0) {
     fscanf(fp, "%s %s %c %s", a.nome, a.cognome, &a.genere, a.nazione);
     for (int i = 0; i < NUM_GIUDICI; i++)
        fscanf(fp, "%f", &a.punteggi[i]);
     fscanf(fp, "%f", &a. gradoDifficolta);

     g.array[k] = a;
     k += 1;
   } // end while

   fclose(fp);
   return g;

} // end acquisisciDaFile
```

Se si continua sul retro di qualche foglio, indicare quale

Soluzione 4(b)

```
void stampa_migliori(gara_t g) {

   if (g.array == NULL || g.numAtleti <= 0) return;

   float maxPunteggio = 0;
   int maxPosizione;
   for (int i = 0; i < g.numAtleti; i++) {

      float punteggio = g.array[i].punteggi[0];
      float punteggioINF = punteggio;
      float punteggioSUP = punteggio;
      for (int j = 1; j < NUM_GIUDICI; j++) {
        if (g.array[i].punteggi[j] < punteggioINF)
          punteggioINF = g.array[i].punteggi[j];
        if (g.array[i].punteggi[j] > punteggioSUP)
          punteggioSUP = g.array[i].punteggi[j];

        punteggio += g.array[i].punteggi[j];
      } // end for
      punteggio -= (punteggioINF+punteggioSUP);
      punteggio *=  g.array[i].gradoDifficolta;

      if (maxPunteggio <= punteggio)
        maxPunteggio = punteggio;
        maxPosizione = i;
      }

   } // end for

   printf("\n Il nome di uno degli atleti con il punteggio ");
    printf("massimo e': %s", g.array[i].nome);


} // end stampa_migliori
```

Se si continua sul retro di qualche foglio, indicare quale