POLITECNICO
MILANO 1863

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA

CORSO DI LAUREA IN INGEGNERIA AEROSPAZIALE

Prof. G. Agosta
Prof. Gerardo Pelosi

Informatica – a.a. 2022/2023– 2º Appello – 9 Febbraio 2023

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.

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:30**

Valore indicativo degli esercizi, voti parziali e voto finale:

Esercizio 1 (4 punti) _____

Esercizio 2 (4 punti) _____

Esercizio 3 (6 punti) _____

Esercizio 4 (8 punti) _____

Esercizio 5 (10 punti) _____

Totale (32 punti) _____

Voto finale _____

Esercizio 1 [4 punti]

La funzione `foo` (vedere codice allegato) ha come obiettivo quello di modificare la stringa `s` rimuovendo l'ultimo carattere se e solo se la differenza numerica tra il primo e l'ultimo carattere della stringa stessa è pari in valore assoluto al valore del parametro `num`. Inoltre, la funzione stampa la differenza calcolata e il parametro `num`.

Ad esempio, se `foo` viene invocata con i parametri `"ASTRA"` e `0`, l'ultima lettera di `ASTRA` deve essere sovrascritta con il carattere terminatore.

Fornendo `"ASTRA"` come stringa iniziale, il `main` fornito dovrebbe quindi restituire come output a terminale il seguente testo:

```
ASTRA
0, 0
ASTR
1, 1
AST
AST
```

La funzione `foo` fornita presenta però due errori di sintassi che ne impediscono la corretta compilazione, e due ulteriori errori logici che ne impediscono il corretto funzionamento.

Trovare e correggere i quattro errori.

```c
#include <stdio.h>
#include <string.h>

int foo(char *s, int num) {
  int lung = strlen(s);
  if (lung <= 1) return 0
  int temp = (int)s[0]-(int)s[lung];
  printf("%d, %d\n,temp, num);
  if (temp == num || temp == -num) {
    s[lung-1] = '0';
    return 1;
  }
  else
    return 0;
}

int main() {
  char str[20];
  int i = 0, flag;
  scanf("%s", str);
  printf("%s\n", str);
  do {
    flag = foo(&str[i], i);
    i += 1;
    printf("%s\n", str);
  } while (flag == 1);

} // end main
```

Se si continua sul retro di qualche foglio, indicare quale

Soluzioni:

```
int foo(char *s, int num) {
  int lung = strlen(s);
  if (lung <= 1) return 0;
  int temp = (int)s[0]-(int)s[lung-1];
  printf("%d, %d\n",temp, num);
  if (temp == num || temp == -num) {
    s[lung-1] = '\0';
    return 1;
  }
  else
    return 0;
}
int main() {
  char str[20];
  int i = 0, flag;
  scanf("%s", str);
  printf("%s\n", str);
  do {
    flag = foo(&str[i], i);
    i += 1;
    printf("%s\n", str);
  } while (flag == 1);
  return 0;
} // end main
```

Esercizio 2 [4 punti]

Implementare una funzione di prototipo

```
char *neg(char *s)
```

che riceve una stringa contenente caratteri `'0'` e `'1'`, interpretata come un numero binario `n` (la cifra più significativa è nella posizione `0`), e restituisce una nuova stringa che contiene il numero `-n` (opposto), espresso sullo stesso numero di bit.

Soluzioni:

```
char *neg(char *s){
  int l=strlen(s);
  int carry=1;
  char *r = malloc(sizeof(char)*(l+1));
  r[l]=0;
  for(int i=l-1; i>=0; i--)
    if (s[i]=='0') {
      if (!carry) r[i]='1';
      else r[i]='0';
    } else {
      if (!carry) r[i]='0';
      else { r[i]='1'; carry=0; }
    }
  return r;
}
```

Esercizio 3 (6 punti)

Scrivere una funzione di prototipo:

```
int isogramma(char *s);
```

che determina se la stringa `s` è un isogramma, ovvero è composta da lettere che ricorrono tutte lo stesso numero di volte.

Ad esempio, la parola "`prosciugante`" è un isogramma, precisamente il più lungo nella lingua italiana.

Le lettere minuscole e maiuscole vengono considerate identiche, quindi, ad esempio, "`Anna`" è un isogramma.

La funzione restituisce la numerosità di ciascuna lettera nell'isogramma, oppure `0` se la parola non è un isogramma.

Soluzioni:

```
int isogramma(char *s){
  int freq[26]={0};
  int f=0;
  for(int i=0; s[i]!='\0'; i++)
    if (s[i]>='A' && s[i]<='Z') s[i]=s[i]-'A'+'a';
  for(int i=0; s[i]!='\0'; i++)
    freq[s[i]-'a']+=1;
  for(int i=0; i<'z'-'a'; i++)
    if (f==0 && freq[i]!=0) f=freq[i];
    else if (freq[i]!=0 && freq[i]!=f) return 0;
  return f;
}
```

Se si continua sul retro di qualche foglio, indicare quale

Esercizio 4 (8 punti)

Data una matrice di interi con R righe e C colonne, scrivere un sottoprogramma che riorganizzi

- gli elementi lungo le righe di indice **dispari**, come sequenze di valori che scandite da sinistra a destra esibiscono in **colonna `[0]` il massimo** valore nelle colonne `0...C-1`, in **colonna `[1]` il minimo** valore nelle colone `1...C-1`, in colonna `[2]` il massimo valore nelle colonne `2...C-1`, in colonna `[3]` il minimo valore nelle colonne `3...C-1` e così via alternando massimi e minimi con lo stesso criterio fino a esaurimento dei valori sulla riga.
- gli elementi lungo le righe di indice **pari**, come sequenze di valori che scandite da sinistra a destra esibiscono **in colonna `[0]` il minimo** valore nelle colone `0...C-1`, **in colonna `[1]` il massimo** valore nelle colonne `1...C-1`, in colonna `[2]` il minimo valore nelle colonne `2...C-1`, in colonna `[3]` il massimo valore nelle colonne `3...C-1` e così via alternando minimi e massimi con lo stesso criterio fino a esaurimento dei valori sulla riga.

Esempio:

	[0]	[1]	[2]	[3]	[4]
[0]	9	6	8	3	7
[1]	4	5	7	1	3
[2]	7	11	0	9	11

3	8	3	7	7
7	1	7	1	3
0	11	0	11	11

Nota:

Nella versione originale del testo, al posto dell'esempio precedente era stato riportato il seguente:

	[0]	[1]	[2]	[3]	[4]
[0]	9	6	8	3	7
[1]	4	5	7	1	3
[2]	7	11	0	9	11

3	9	6	8	7
7	1	5	3	4
0	11	7	11	9

Il cui testo sarebbe consistito in:

- Per le righe di indice dispari:
    - per le colonne di indice pari: inserire i valori in ordine decrescente partendo dal massimo.
    - per le colonne di indice dispari: inserire i valori in ordine crescente partendo dal minimo.

Se si continua sul retro di qualche foglio, indicare quale

- Per le righe di indice pari:
    - per le colonne di indice pari: inserire i valori in ordine crescente partendo dal minimo.
    - per le colonne di indice dispari: inserire i valori in ordine decrescente partendo dal massimo.

Ai fini della correzione, sono stati considerate valide entrambe le versioni.

Suggerimento:
organizzare il codice prevedendo almeno due procedure aggiuntive per il riordino di una riga, che possano essere invocate come segue:

```
riorganizza_crescente(&M[i][0], colonne);   // i == valore pari
riorganizza_decrescente(&M[i][0], colonne); // i == valore dispari
```

Soluzioni:

Versione A:

```
void riorganizza_crescente(int vett[], int n) {
    int massimo, minimo;
    int posMassimo, posMinimo;

    for (int i = 0; i < n; i++) {
        massimo = minimo = vett[i];
        posMassimo = posMinimo = i;
        for (int k = i+1; k < n; k++) {
            if (vett[k] > massimo) { posMassimo = k; massimo = vett[k]; }
            if (vett[k] < minimo) { posMinimo = k; minimo = vett[k]; }
        }
        if (i%2 == 0) vett[i] = vett[posMinimo];
        else vett[i] = vett[posMassimo];
    }
}

void riorganizza_decrescente(int vett[], int n) {
    int massimo, minimo;
    int posMassimo, posMinimo;

    for (int i = 0; i < n; i++) {
        massimo = minimo = vett[i];
        posMassimo = posMinimo = i;
        for (int k = i+1; k < n; k++) {
            if (vett[k] > massimo) { posMassimo = k; massimo = vett[k]; }
            if (vett[k] < minimo) { posMinimo = k; minimo = vett[k]; }
        }
        if (i%2 == 0) vett[i] = vett[posMassimo];
        else vett[i] = vett[posMinimo];
    }
}

void riorganizzazione(int *mat, int r, int c) {
    for (int i = 0; i < r; i++) {
        if (i%2 == 0) riorganizza_crescente(mat + i * c, c);
        else riorganizza_decrescente(mat + i * c, c);
    }
}
```

Versione B:

```c
int maxpos(int *a, int l){
    int i, max=0;
    for(i=1; i<l; i++)
        if (a[i]>a[max])
            max=i;
    return max;
}

void scambia(int *x, int *y){
    int tmp=*x;
    *x=*y;
    *y=tmp;
}

int *ordina(int *a, int l){
    int *r=malloc(sizeof(int)*l);
    for(int i=0; i<l; i++) r[i]=a[i];
    for(int i=0; i<l; i++) scambia(&r[i+maxpos(&r[i],l-i)],&r[i]);
    return r;
}

void riorganizza_crescente(int R[], int c){
    int *r=ordina(R, c);
    for(int i=0; i<c; i++) { /* Soluzione acrobatica :-) */
        R[i]=(i%2?r[i/2]:r[c-i/2-1]);
    }
    free(r);
}

void riorganizza_decrescente(int R[], int c){
    int *r=ordina(R, c);
    int j=c,k=0;
    for(int i=0; i<c; i++) { /* Soluzione piu' semplice */
        if(i%2==0)
            R[i]=r[k++];
        else
            R[i]=r[--j];
    }
    free(r);
}

void riorganizzazione(int *mat, int r, int c) {
    for (int i = 0; i < r; i++) {
        if (i%2 == 0) riorganizza_crescente(mat + i * c, c);
        else riorganizza_decrescente(mat + i * c, c);
    }
}
```

Esercizio 5 (10 punti)

- **Parte A (5 punti)**

  Realizzare una funzione di prototipo

  `tPersona *lettura_persone(int *n);`

  che legga `n` righe, ciascuna costituita da un nome, un cognome (ciascuno rappresentato da una sequenza di caratteri priva di spazi) e una data di nascita nel formato `DD/MM/YYYY`, scartando le righe per le quali la data di nascita è malformata (ovvero il giorno o il mese non sono compresi negli intervalli corretti, o la data è nel futuro; si assuma inoltre che gli anni di nascita siano positivi). Al fine di tali controlli, non è necessario verificare che il 29 Febbraio cada effettivamente in un anno bisestile. In caso di errore, la funzione `lettura_persone` deve chiamare la funzione `stampa_errore` data, passando ad essa l'opportuno codice d'errore (vedere il tipo enumerativo `tControlloData`).

  La funzione deve restituire una struttura dati dinamicamente allocata contenente i dati letti.

  Il parametro `n` contiene all'atto della chiamata il numero di righe da leggere, ma all'atto della chiusura della funzione deve contenere il numero di righe valide lette.

- **Parte B (5 punti)**

  Realizzare inoltre una funzione di prototipo

  `int distanza(tPersona *data, int n);`

  che riceva una struttura di tipo `tPersona` contenente i dati di `n` persone, e calcoli la distanza (espressa in anni) fra l'anno di nascita della persona più vecchia e l'anno di nascita della persona più giovane.

Codice noto (continua nella pagina seguente):

```c
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>

#define NMAX 128

typedef enum {
  OK,
  GIORNOERRATO,
  MESEERRATO,
  ANNOERRATO,
  DATAFUTURA,
} tControlloData;

typedef enum {
  NONE, GEN, FEB, MAR, APR, MAG, GIU, LUG, AGO, SET, OTT, NOV, DIC
} tMese;

typedef struct _data {
  int giorno;
  tMese mese;
  int anno;
} tData;

const tData oggi = { 9, 2, 2023 };
```

```c
typedef struct _tPersona {
  char nome[NMAX];
  char cognome[NMAX];
  tData data_nascita;
} tPersona;

void stampaerrore(tControlloData c){
  printf("errore: %d\n", c);
}

tPersona *lettura_persone(int *n);
int distanza(tPersona *data, int n);

int main(){
  int n;
  tPersona *dati=NULL;
  scanf("%d", &n);
  dati=lettura_persone(&n);
  printf("n: %d\n", n);
  printf("distanza: %d\n", distanza(dati, n));
  return 0;
}
```

Esempio:

stdin
4 Mario Rossi 19/01/1980 Luigi Verdi 02/13/1961 Alice Bianchi 09/02/1995 Sofia Gatti 25/08/2034

stdout
errore: 2 errore: 4 n: 2 distanza: 15

Soluzioni:

```
tControlloData controlla_data(tData d){
  if (d.giorno<1 || d.giorno>31) return GIORNOERRATO;
  if (d.mese<GEN || d.mese>DIC) return MESEERRATO;
  if (d.anno>oggi.anno ||
      (d.anno==oggi.anno && d.mese > oggi.mese) ||
      (d.anno==oggi.anno && d.mese==oggi.mese && d.giorno > oggi.giorno))
    return DATAFUTURA;
  if (d.anno<=0) return ANNOERRATO;
  switch (d.mese) {
    case APR :
    case GIU :
    case SET :
    case NOV : if (d.giorno>30) return GIORNOERRATO;
    case FEB : if (d.giorno>29) return GIORNOERRATO;
  }
  return OK;
}

tPersona *lettura_persone(int *n){
  tPersona *res=malloc(sizeof(tPersona)**n);
  tPersona r;
  int j=0;
  for(int i=0; i<*n; i++) {
    tControlloData c;
    scanf("%s %s %d/%u/%d", r.nome, r.cognome, &r.data_nascita.giorno,
          &r.data_nascita.mese, &r.data_nascita.anno);
    c = controlla_data(r.data_nascita);
    if (c==OK) {
      res[j]=r;
      j++;
    } else {
      stampaerrore(c);
    }
  }
  *n=j;
  return res;
}

int massimo(tPersona *persona, int n){
  int amax=0;
  for(int i=0; i<n; i++)
    if (persona[i].data_nascita.anno>amax) amax=persona[i].data_nascita.anno;
  return amax;
}

int minimo(tPersona *persona, int n){
  int amin=oggi.anno;
  for(int i=0; i<n; i++)
    if (persona[i].data_nascita.anno<amin) amin=persona[i].data_nascita.anno;
  return amin;
}

int distanza(tPersona *persona, int n){
  return massimo(persona,n)-minimo(persona,n);
}
```