POLITECNICO
MILANO 1863

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA

CORSO DI LAUREA IN INGEGNERIA AEROSPAZIALE

Prof. Gerardo Pelosi

Informatica – a.a. 2020/2021– 3º Appello – 9 Settembre 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:15**

Valore indicativo degli esercizi, voti parziali e voto finale:

Esercizio 1 (7 punti) _____

Esercizio 2 (8 punti) _____

Esercizio 3 (10 punti) _____

Esercizio 4 (8 punti) _____

Totale (33 punti) _____

Voto finale _____

Esercizio 1 Codifiche numeriche e algebra di Boole [7 punti]

(a) Si costruisca la tabella di verità della seguente espressione Booleana. [2 punti]

F(A,B,C) = A and (B or not C) or B or not (A and not (C or not B))

(b) Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due entrambi i numeri A = **–62**dec e B = **79**dec, li si converta in complemento a due, se ne calcolino la somma (A + B) e la differenza (A – B) in complemento a due e si indichi se si genera riporto dalla colonna dei bit più significativi e se si verifica overflow. [4 punti]

(c) Si esprima il numero esadecimale $D1EC1_{hex}$ in base due e in base otto [1 punto]

Soluzione

A	B	C	$A\cdot(B+\overline{C}) + B + \overline{(A\cdot(C+\overline{B})}$	F(A,B,C)
0	0	0	0·(0+1)+0+not(0·not(0+1)) = *1*	1
0	0	1	0·(0+0)+0+not(0·not(1+1)) = *1*	1
0	1	0	0·(1+1)+1+not(0·not(0+0)) = *1*	1
0	1	1	0·(1+0)+1+not(0·not(1+0)) = *1*	1
1	0	0	1·(0+1)+0+not(1·not(0+1)) = *1*	1
1	0	1	1·(0+0)+0+not(1·not(1+1)) = *1*	1
1	1	0	1·(1+1)+1+not(1·not(0+0)) = *1*	1
1	1	1	1·(1+0)+1+not(1·not(1+0)) = *1*	1

Si osserva che nessuno dei due numeri A, B è una potenza di due positiva dunque è possibile applicae la formula:

$n_{c2} = 1+\min\{ \text{ceil}(\log_2(67)), \text{ceil}(\log_2(79)) \} = 1+\min\{7, 7\} = 8$ bit

A = $\mathbf{-62}_{dec} = ... = 11000010_{c2}$

B = $\mathbf{79}_{dec} = ... = 01001111_{c2}$

A + B = ... = [1] 00010001_{c2} SI, riporto oltre la cifra più significativa.
NO, *overflow* (perché addendi discordi)

A = $\mathbf{-62}_{dec} = ... = 11000010_{c2}$

-B = $\mathbf{-79}_{dec} = ... = 10110001_{c2}$

A + (-B) = ... =[1] 01110011_{c2} SI, riporto oltre la cifra più significativa.
SI *overflow* (perché addendi concordi e somma discorde)

$D1EC1_{hex} = 1101\ 0001\ 1110\ 1100\ 0001_{bin} = 3217301_{Oct}$

Esercizio 2 (8 punti)

Si consideri la simulazione del movimento di un gatto su una mappa rappresentata come una matrice quadrata con lato di lunghezza N celle. Ogni cella della matrice contiene un numero non negativo e codificato in virgola mobile per indicare la quota (distanza dal suolo) del quadratino che le corrisponde sulla mappa. Date le coordinate della posizione attuale del gatto sulla matrice, quest'ultimo si sposta in una cella adiacente, solo se questa memorizza un valore di quota che è il massimo tra quella attuale e quelle memorizzate in ciascuna delle posizioni adiacenti;

- se tutti i quadratini adiacenti alla posizione attuale memorizzano una quota inferiore o uguale a quella attuale, il gatto si ferma;
- se in più di una delle celle adiacenti alla posizione attuale è memorizzato il valore di quota massimo, il gatto può muoversi su una qualunque di esse.

Date le seguenti definizioni:

```
#define N ...

typedef struct coordinate { int r;
                            int c;
        } coord;

int controllo(int r, int c) {
   if ( r < 0  || r > N || c < 0 || c > N )
     return 0;                              // coordinate non valide
   return 1;                                // coordinate valide
}
```

(a) Si scriva un sottoprogramma C, `...massimoAdiacenti(...)` che abbia come parametri una matrice quadrata di lato N e le coordinate di una cella della matrice, e che restituisca le coordinate della cella contenete il valore massimo tra quelli memorizzati nella cella passata come parametro e nelle celle adiacenti. *[4 punti]*

Soluzione

```
coord massimoAdiacenti(double mappa[][N], coord in) {

   coord ret = {-1, -1};

   if (controllo(in.r, in.c) == 0)
     return ret;

   double massimo = mappa[in.r][in.c];
   int dx, dy, rmax, cmax;

   for (dx = -1; dx < 2; ++dx) {
     for (dy = -1; dy < 2; ++dy) {
        if ( controllo(in.r+dx, in.c+dy) == 1 &&
             massimo < mappa[in.r+dx][in.c+dy]   ) {
          massimo = mappa[in.r+dx][in.c+dy];
          rmax = in.r+dx;
          cmax = in.c+dy;
        } // end if
     } // end for
   } // end for
   ret.r = rmax;
   ret.c = cmax;
   return ret;
}   // end massimoAdiacenti
```

Se si continua sul retro di qualche foglio, indicare quale

(b) Si scriva un sottoprogramma C, `...Spostamento(...)` che abbia come parametri una matrice quadrata di lato `N` e le coordinate della posizione iniziale del gatto, e che restituisca le coordinate della posizione sulla mappa dalla quale il gatto non riesce più a spostarsi, oppure le coordinate `(-1,-1)` per indicare che la posizione iniziale non è valida. *[4 punti]*

Esempio:
Con `N == 4`, coordinate iniziali `(0,0)`, e la matrice riportata in figura, le coordinate dell'ultima cella raggiunta sono: `(3,1)`

	[0]	**[1]**	**[2]**	**[3]**
[0]	***1.7***	2.2	3.9	9.7
[1]	2.3	***5.6***	4.2	3.2
[2]	4.8	5.0	***5.9***	3.2
[3]	4.8	***8.5***	6.7	2.9

Soluzione

```
coord Spostamento(double mappa[][N], coord in) {

    coord ret = {-1, -1};
    if ( controllo(in.r, in.c) == 0 )
      return ret;

    coord co = in;
    coord coordMax;
    int avanti = 1;

    do {
         coordMax = massimoAdiacenti(mappa, co);
         if ( coordMax.r == co.r && coordMax.c == co.c  )
           avanti = 0;
         else
           co = coordMax;
    } while ( avanti == 1 );

    return co;

} // end Spostamento
```

Esercizio 3 (10 punti)

Due parole *p, q* si definiscono allacciabili se un suffisso proprio ***s*** di *p* è anche prefisso proprio di *q*, cioè hanno la forma *p=w1**s*** *q=**s**w2* (dove ***s*** è proprio se è lungo almeno 2 lettere ma più corto di *p* e di *q*).

Esempi di parole allacciabili:

*(oca, carina) → o**ca**rina* *(isola, lamento) → iso**la**mento*
*(bugiardi, giardino) → bu**giardi**no(dei farmaci)* *(spora, radici) → spo**ra**dici*
*(imposta, stazione) → impo**sta**zione* *(violoncello, cellophane) → violon**cello**phane*

Malgrado le coppie di parole allacciabili più interessanti siano quelle in cui la parola w1sw2 (ottenuta fattorizzando il suf-/pref-isso) è una parola di senso compiuto, anche l'orrenda coppia (violoncello, cellophane) rispetta la definizione.

Esempi di coppie non allacciabili:

*(violon**cell**o, **cell**ulare),* *(**forma**, **forma**zione),*
*(co**raggio**, **raggio**),* *(corp**o**, **o**razione),* *(trogolo, zangola)*

(a) Ipotizzando di disporre di una funzione `int misuraoverlap(char *p, char *q)` che restituisce la lunghezza del massimo suf-/pre-fisso proprio in comune tra *p* e *q* (cioè la lunghezza di *s,* se *p=w1s*, *q=sw2*), si codifichi in C la funzione `...allaccia(...)` che riceve due parole come parametri e restituisce una nuova stringa (allocata dinamicamente) contenente la parola ottenuta dall'allacciatura che fattorizza la massima sovrapposizione, oppure NULL se le parole non sono allacciabili [5 punti]

N.B: è consentito/raccomandato far uso delle funzioni di libreria in `<string.h>` (`strlen, strcpy, strcat`) e in `<stdlib.h>` (`malloc`), e/o definire dei proprii sottoprogrammi.

Soluzione

Misuriamo l'overlap (la lunghezza della massima s) con la funzione opportuna, e se è maggiore di 2 e minore delal lunghezza di entrambe le parole allochiamo una stringa di dimensione opportuna.

```
char* allaccia( char * p, char * q ) {

     char* r = NULL;
     int ov = misuraoverlap(p,q), lp = strlen(p), lq = strlen(q);

     if (ov >= 2) {
       r = (char *) malloc(lp + lq - ov + 1);
       strcpy(r, p);
       strcat(r, q+ov);
      } // end if

     return r;

} // end allaccia
```

Se si continua sul retro di qualche foglio, indicare quale

(b) Si codifichi in C la funzione **`int`** `misuraoverlap(`**`char`** `*p,` **`char`** `*q)`
Si complementi la soluzione spieghando in modo chiaro, sintetico e preciso come funziona l'algoritmo scelto per `misuraoverlap()`, e in particolare come garantisce che la sottostringa **s** sia "massima" (cioè non esista un **s'** > **s**) [5 punti]

Soluzione

*L'algoritmo inizializza max_ov alla lunghezza della stringa più corta meno un carattere (che è la massima dimensione teorica per le sovrapposizione proprie) e prova, per ogni valore intero in [2..max_ov] (in ordine decrescente), a verificare se esiste una **s** di tale lunghezza, fermandosi al primo valore trovato o restituendo 0 se non ne trova. Ciò garantisce che il valore trovato sia il più alto che verifica la proprietà.*

```
int misuraoverlap(char* p, char* q) {

   int max_ov = strlen(p)-1;

   if (max_ov >= strlen(q))
     max_ov = strlen(q)-1;

   for ( ; max_ov >= 2 ; max_ov--)
      if (SovrapponibiliConNcaratteri(p, q, max_ov))
        return max_ov;

   return 0;

} // end misuraoverlap


int SovrapponibiliConNcaratteri(char* p, char* q, int n) {

   for (int i = 0; i < n; i++)
      if (p[strlen(p)- n+i] != q[i])
        return 0;

   return 1;

} // end SovrapponibiliConNcaratteri
```

Se si continua sul retro di qualche foglio, indicare quale

Esercizio 4 (8 punti)

Si consideri il seguente programma:

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

#define MAXSTRLENNOME (29+1)
#define MAXSTRLENEX (99+1)
#define MAXEXR 10

typedef struct {
   char nomeEsercizio[MAXSTRLENEX];
   int peso;
   int ripetizioni;
} esercizio;

typedef struct {
   char nomeAtleta[MAXSTRLENNOME];
   esercizio ex[MAXEXR];
   int numero_esercizi; // numero indicante il numero di celle
                        // del vettore ex[...] riempite effettivamente.
} scheda;

scheda* lettura_dati(char* filename);

int soglia(scheda* s, float peso_soglia);

int scrittura_dati(char* filename, scheda* s);

int main() {
   char* fname = "scheda.csv";
   scheda* s = lettura_dati(fname);
   int n2 = soglia(s, 64);
   int r = scrittura_dati("scheda2.csv", s);
   free(s);
   return r;
}
```

In cui il *file* chiamato `scheda.csv` contiene nella prima riga il nome di una persona (stringa di al più 29 caratteri senza caratteri di spaziatura), e in ciascuna delle righe successive una stringa che rappresenta il nome di un esercizio (senza caratteri di spaziatura), un numero intero che rappresenta il peso impiegato nell'esercizio e infine un altro numero intero che rappresenta il numero di ripetizioni dell'esercizio. Le tre informazioni su ciascun rigo del file, eccetto il primo rigo, sono separate da un carattere di spazio `' '`.

Esempio di contenuto del *file* `scheda.csv` :

```
Paolino_Paperino
Panca_piana 40 10
Panca_inclinata 30 12
Aperture_laterali 8 12
Tricipiti_alla_poliercolina 12 12
```

Se si continua sul retro di qualche foglio, indicare quale

(a) Scrivere una funzione di prototipo

```
int soglia(scheda* s, float pesoSoglia)
```

che riceve in ingresso una scheda (con passaggio parametri per indirizzo) e un valore `pesoSoglia` usato per rimodulare i pesi di tutti gli esercizi nella scheda in modo tale che il *peso_totale* spostato per ogni esercizio non ecceda il valore di soglia.

Esempio:
se la scheda contiene un esercizio descritto dal rigo del *file*:
`Panca_piana 24 12`
e `pesoSoglia` è impostato a 64 (kg),
l'esercizio nella scheda sarà modificato impostando come peso da usare in ogni ripetizione la parte intera di pesoSoglia / *numero_di_ripetizioni*, ottenendo `Panca_piana 5 12`

Soluzione

```
int soglia(scheda* s, float pesoSoglia) {
   if (s == NULL) return 0;
   int rimodulatoAlmenoUnaVolta = 0;

   for (int i = 0; i < s->numero_esercizi; i++) {
      if (s->ex[i].peso * s->ex[i].ripetizioni > pesoSoglia) {
        s->ex[i].peso = pesoSoglia / s->ex[i].ripetizioni;
        rimodulatoAlmenoUnaVolta = 1;
      } // end if
   } // end for

   return rimodulatoAlmenoUnaVolta;
} // end soglia
```

(b) Scrivere una funzione

```
void scrittura_dati(char* nomefile, scheda* s)
```

avente come secondo parametro una variabile di tipo `scheda` (passata per indirizzo) che scriva il contenuto della scheda sul *file* il cui nome è passato come primo parametro, nel formato indicato per questo esercizio.

Soluzione

```
void scrittura_dati(char* nomefile, scheda* s) {

    FILE* fp = fopen(nomefile, "w");

    if (fp == NULL || s == NULL) {
      fprintf(stderr, "\n Errore in creazione file o in var. scheda!\n");
      return;
    } // end if

    fprintf(fp,"%s\n", s->nomeAtleta);

    for (int i = 0; i < s->numero_esercizi; i++) {
       fprintf(fp, "%s %d %d\n", s->ex[i].nomeEsercizio,
                                 s->ex[i].peso,
                                 s->ex[i].ripetizioni);
    } // end for
    fclose(fp);

} // end scrittura_dati
```

Se si continua sul retro di qualche foglio, indicare quale