CORSO DI LAUREA IN INGEGNERIA AEROSPAZIALE

Prof. G. Agosta
Prof. Gerardo Pelosi

Informatica – a.a.2022/2023 – 4to Appello – 17 Luglio 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	(6 punti)	_____
Esercizio 2	(4 punti)	_____
Esercizio 3	(10 punti)	_____
Esercizio 4	(12 punti)	_____
Totale	(32 punti)	_____

Voto finale

Se si continua sul retro di qualche foglio, indicare quale

Esercizio 1 [6 punti]

Scrivere un programma che legga da terminale due sequenze di caratteri (di lunghezza compresa tra 1 e 8) da interpretare come cifre in base 7.
Ciascuna sequenza è conclusa da un carattere di spaziatura.

Il programma calcola la differenza fra i due valori, e stampa il valore come stringa in rappresentazione binaria in complemento a due, usando il numero minimo di bit necessario a rappresentare tale risultato.

Nel caso le stringhe non rispettino la specifica, il programma stamperà la stringa "`errore 1`" o "`errore 2`" sul terminale di errore a seconda che la prima stringa specificata in modo scorretto sia la prima o la seconda.
In entrambi i casi, il programma restituisce il valore 0 senza proseguire.
Tutte le stampe devono terminare con un carattere di "a capo".

Esempi:

Input
```
4 7
```

Output
```
errore 2
```

Input
```
213 123
```

Output
```
0101010
```

Input
```
c
```

Output
```
errore 1
```

Se si continua sul retro di qualche foglio, indicare quale

Soluzione:

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

int b7c(char c) {
    if (c >= '0' && c <= '6') {
        return c - '0';
    }

    return -1;
}

int b7toint(char *s, int *n) {
    int c;
    *n = 0;

    for (int i = 0; s[i] != '\0'; i++) {
        c = b7c(s[i]);

        if (c<0) {
            return -1;
        }

        *n = *n * 7 + c;
    }

    return 0;
}

void print_bits(int m) {
    unsigned int *n = &m;
    int l;

    if (m == 0) {
        l = 0;
    } else if (m < 0) {
        l = ceil(log2(-m));
    } else {
        l = n ? 1 + log2(m) : 1;
    }

    char *s = malloc(sizeof(char) * (l + 1));

    if (!s) return;

    memset(s, '0', l + 1);

    for (int i = 0; *n && i<=l; i++) {
        s[l-i] = *n % 2 + '0';
        *n = *n / 2;
    }

    printf("%s\n", s);
    free(s);
}
```

Se si continua sul retro di qualche foglio, indicare quale

```c
int main() {
    char s[9];
    int n1 = 0, n2 = 0;
    int l;

    scanf("%s", s);
    l = strlen(s);

    if (l < 1 || l > 8) {
        return 0;
    }

    if (b7toint(s, &n1)) {
        printf("errore 1\n");
        return 0;
    }

    scanf("%s", s);
    l = strlen(s);

    if (l < 1 || l > 8) {
        return 0;
    }

    if (b7toint(s, &n2)) {
        printf("errore 2\n");
        return 0;
    }

    print_bits(n1 - n2);
    return 0;
}
```

Se si continua sul retro di qualche foglio, indicare quale

Esercizio 2 [4 punti]

Il seguente programma calcola il numero di mode (ovvero di valore più frequenti) in una sequenza multimodale di caratteri comprensivi di spazi e terminata da `EOF`, a capo o fine stringa (considerati non parte della sequenza).
Ad esempio, data la sequenza `"abcaadded"` il numero di mode è 2 (ovvero la sequenza è bimodale).

Il programma deve stampare a terminale il numero di occorrenze di ogni carattere, preceduto dalla rappresentazione esadecimale del carattere stesso e da un carattere `':'`

Esempi:

Input	**Output**

`assds`	`\x61:1`
	`\x64:1`
	`\x73:3`
	`#mode:1`

Input	**Output**

`a a a a b`	`\x20:4`
	`\x61:4`
	`\x62:1`
	`#mode:2`

Vi sono però alcuni errori, sia di logica che di sintassi, che impediscono il corretto funzionamento. Risolvere tali errori.

```
#inclde <stdio.h>

int main(){
  char c=0;
  int f[256]={0}, mode=0, n=0;
  while(1){
    c=fgetc(stderr);
    if (c==EOF&&c=='\n'&&c=='\0') break;
    f[c]+=1;
  }
  for(int i=0; i<n; i++){
    if (f[i]=0) printf("\x%02x:%d\n",i, f[i]);
    if (f[i]>mode) mode=f[i];
  }
  for(int i=0; i<n; i++){
    if (f[i]==mode) m++;
  }
  printf("#mode:%d\n", n);
}
```

Se si continua sul retro di qualche foglio, indicare quale

Soluzione:

```c
#include <stdio.h>

int main() {
    int c;
    int f[256] = {0}, mode = 0, n = 0;

    while (1) {
        c = fgetc(stdin);

        if (c == EOF || c == '\n' || c == '\0') {
            break;
        }

        f[c] += 1;
    }

    for (int i = 0; i < 256; i++) {
        if (f[i] != 0) {
            printf("\\x%02x:%d\n", i, f[i]);
        }

        if (f[i] > mode) {
            mode = f[i];
        }
    }

    for (int i = 0; i < 256; i++) {
        if (mode != 0 && f[i] == mode) {
            n++;
        }
    }

    printf("#mode:%d\n", n);
    return 0;
}
```

Se si continua sul retro di qualche foglio, indicare quale

Esercizio 3 [10 punti]

Il biscarto è un gioco enigmistico che pone in relazione tre stringhe, ottenendo la terza dalla concatenazione delle prime due, da cui sia stata tolta una sottostringa comune.

Si progetti un programma di grado di determinare se tre stringhe lette da terminale costituiscono un biscarto.

In particolare, la funzione `biscarto` dovrà restituire `-1` se la terna non costituisce un biscarto, e in caso contrario un numero intero positivo che rappresenti la posizione nella terza stringa della prima lettera derivante dalla seconda stringa.

Esempi:

Biscarto (semplice): pavoni / covo = panico (scarto: "vo")

Biscarto centrale: l'arco / l'ombra = la colomba (scarto: "r")

Biscarto iniziale: la claque / l'acrimonia = la querimonia (scarto: "lac")

Biscarto finale: minio / golfo = minigolf (scarto: "o")

Si noti che negli esempi vengono riportati anche gli apostrofi, accenti e spazi che nei casi di test si considerano già rimossi.

Ad esempio, un caso di test per il biscarto iniziale dell'esempio sarà:

Input	**Output**
`laclaque lacrimonia laquerimonia`	`5`

Si consideri noto il seguente codice:

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

int biscarto(char *a, char *b, char *c);

int main(){
  char a[128], b[128], c[128];
  scanf("%s %s %s", a,b,c);
  printf("%d\n", biscarto(a,b,c));
  return 0;
}
```

Se si continua sul retro di qualche foglio, indicare quale

Soluzione:

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

int scarto(char *s, char *t, int p, int l);

int biscarto(char *a, char *b, char *c);

int main() {
    char a[128], b[128], c[128];
    scanf("%s%s%s", a, b, c);
    printf("%d\n", biscarto(a, b, c));
    return 0;
}

int scarto(char *s, char *t, int p, int l) {
    if (strlen(t)+l!=strlen(s)) return 0;
    for(int i=0, j=0; i<strlen(t); i++,j++) {
        if (j==p) j+=l;
        if (s[j]!=t[i]) return 0;
    }
    return 1;
}

int biscarto(char *a, char *b, char *c) {
    int la = strlen(a);
    int lb = strlen(b);
    int lc = strlen(c);
    int p1, p2;
    int i;

    if ((la + lb - lc) % 2) {
        // nessun biscarto perché i residui non hanno la lunghezza giusta
        return -1;
    }

    int lscarto = (la + lb - lc) / 2; // lunghezza dello scarto per ciascuna stringa
    char *cdup = strdup(c);

    for(i = 1; i < lc - 1; i++) { // proviamo la posizione i come punto di sezione
della stringa c
        cdup[i] = '\0';
        int r1 = 0, r2 = 0;

        for (p1=0; p1 <= la - lscarto; p1++) { // cerchiamo una posizione p1 per lo
scarto in a
            r1 = scarto(a, cdup, p1, lscarto);

            if (!r1) {
                continue;
            }

            for(p2=0; p2 <= lb - lscarto; p2++){ // cerchiamo una posizione p2 per lo
scarto in b
                r2 = scarto(b, &c[i], p2, lscarto);

                if (!r2) {
```

Se si continua sul retro di qualche foglio, indicare quale

```
                    continue;
                }

                if (strncmp(&a[p1], &b[p2], lscarto) == 0){
                    free(cdup);
                    return i;
                }
            }
        }

        strcpy(cdup,c);
    }

    free(cdup);
    return -1;
}
```

Se si continua sul retro di qualche foglio, indicare quale

Esercizio 4 [12 punti]

I dati degli atleti di una società sportiva sono classificati in un file, che contiene per ogni riga la descrizione di un atleta, costituita dal suo nome (una stringa di caratteri alfabetici di al più 31 elementi), e da quattro numeri compresi tra 0 e 20, che rappresentano la valutazione, nell'ordine, della forza, agilità, velocità e resistenza dell'atleta.

Ad esempio, un file di questo tipo potrebbe contenere i dati:

```
tizio      14 15 6  8
caio       10 12 10 12
sempronio  12 14 18 11
```

Per costituire una squadra, è necessario che gli atleti siano dotati di caratteristiche compatibili fra loro. A tale scopo, si considera che gli atleti di una squadra siano compatibili se, per ciascuna delle loro caratteristiche, non vi è una distanza fra due atleti pari o superiore a 5.

Impiegando il `main` ed i tipi di strutture dati forniti, realizzare un programma che:

a. Carica da file i dati in una apposita struttura dati di tipo "squadra" (funzione "`carica_dati`"); **[5 punti]**
b. Verifica che una squadra sia compatibile, restituendo `1` in tale caso e `0` in caso contrario (funzione "`verifica`"); **[4 punti]**
c. Nel caso in cui vi siano più di sei atleti nella società, determina se sia possibile comporre una squadra di 5 elementi compatibili (funzione "`esiste_squadra`"). **[3 punti]**

È possibile verificare le tre funzionalità separatamente sul server, impiegando i tre set di test forniti.

Codice fornito:

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

typedef enum {
    FORZA,
    AGILITA,
    VELOCITA,
    RESISTENZA
} caratteristiche_t;

typedef struct {
    char nome[32];
    unsigned short caratteristiche[4]; /* Si possono usare gli elementi di carat-
teristiche_t per indicizzare questo array */
} persona_t;

typedef struct {
    persona_t *elementi; /* Contiene i dati degli atleti */
    unsigned numero;     /* Numero di atleti nel campo elementi */
} squadra_t;

squadra_t carica_dati(char *nome_file);
int verifica(squadra_t s);
int esiste_squadra(squadra_t s);


int main() {
```

Se si continua sul retro di qualche foglio, indicare quale

```
    char nomef[100];
    scanf("%s", nomef);
    squadra_t s = carica_dati(nomef);
    printf("%d\n", s.numero);
    if (s.numero > 1 && s.numero < 5) printf("%d\n", verifica(s));
    if (s.numero > 5) printf("%d\n", esiste_squadra(s));
}
```

Se si continua sul retro di qualche foglio, indicare quale

Soluzione:

```
squadra_t carica_dati(char *nome_file){
    squadra_t r = {0};
    FILE *fin = fopen(nome_file, "r");

    if (!fin) {
        return r;
    }

    persona_t tmp;
    int i;

    if (!fin) {
        fclose(fin);
        return r;
    }

    // Conta il numero di righe
    while (fscanf(fin, "%s%hu%hu%hu%hu",
                  tmp.nome,
                  &tmp.caratteristiche[FORZA],
                  &tmp.caratteristiche[AGILITA],
                  &tmp.caratteristiche[VELOCITA],
                  &tmp.caratteristiche[RESISTENZA]) == 5) {
        r.numero++;
    }

    rewind(fin);
    r.elementi = malloc(sizeof(persona_t) * r.numero);

    if (!r.elementi) {
        r.elementi = 0;
        fclose(fin);
        return r;
    }

    i = 0;

    // Leggi gli elementi
    while (fscanf(fin, "%s%hu%hu%hu%hu", r.elementi[i].nome,
           &r.elementi[i].caratteristiche[FORZA],
           &r.elementi[i].caratteristiche[AGILITA],
           &r.elementi[i].caratteristiche[VELOCITA],
           &r.elementi[i].caratteristiche[RESISTENZA]) == 5) {
        i++;
    }

    fclose(fin);
    return r;
}

int verifica(squadra_t s) {
    unsigned short max[4] = {0};
    unsigned short min[4] = {20, 20, 20, 20};
    int c, p, r = 1;

    for (c = FORZA; c <= RESISTENZA; c++) {
        for(p=0; p < s.numero; p++){
            if (s.elementi[p].caratteristiche[c]>max[c])
```

Se si continua sul retro di qualche foglio, indicare quale

```
                max[c] = s.elementi[p].caratteristiche[c];

            if (s.elementi[p].caratteristiche[c]<min[c])
                min[c] = s.elementi[p].caratteristiche[c];
        }
    }

    for (c = FORZA; c <= RESISTENZA; c++) {
        if (max[c] - min[c] >= 5)
            r = 0;
    }

    return r;
}

int esiste_squadra(squadra_t s) {
    squadra_t r;
    r.numero = 5;
    r.elementi = malloc(sizeof(persona_t) * 5);
    int i, j, k, l, m;

    for (i = 0; i < s.numero; i++) {
        r.elementi[0] = s.elementi[i];

        for (j = i + 1; j < s.numero; j++) {
            r.elementi[1] = s.elementi[j];

            for (k = j + 1; k < s.numero; k++) {
                r.elementi[2] = s.elementi[k];

                for (l = k + 1; l < s.numero; l++) {
                    r.elementi[3] = s.elementi[l];

                    for (m = l + 1; m < s.numero; m++) {
                        r.elementi[4] = s.elementi[m];

                        if (verifica(r)) {
                            return 1;
                        }
                    }
                }
            }
        }
    }

    return 0;
}
```