Concorso di intelligenza

In un concorso di intelligenza, N giudici esprimono il loro giudizio su K candidati (N e K sono interi inseriti dall'utente tramite tastiera, rispettivamente minori di 10 e 100). Il giudizio è un valore numerico tra 0 e 5.

Si scriva un programma in linguaggio C che, dopo aver acquisito da tastiera i giudizi espressi da ciascun giudice per ciascun candidato, determini:

1) il candidato più intelligente (i.e. con somma dei giudizi massima)
2) il giudice più severo (i.e. con somma dei giudizi minima)

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

#define MAXK 100 /* max n. candidati */
#define MAXN 10  /* max n. giudici */

int main(void)
{
    int voti[MAXK][MAXN] ;
    int tot[MAXK] ; /* somma dei voti per ogni candidato */
    int totg[MAXN] ; /* somma dei voti di ogni giudice */
    int K, N ;
    int i, j ;
    int min, max, posmin, posmax ;

    printf("Quanti candidati ci sono? ");
    scanf("%d", &K) ;

    printf("Quanti giudici ci sono? ");
    scanf("%d", &N) ;

    for (i=0; i<K; i++)
    {
        printf("Immettere i giudizi per il candidato %d\n", i+1);

        for (j=0; j<N; j++)
        {
            printf("Giudice %d, cosa pensi del candidato %d? ",
                    j+1, i+1 );
            scanf("%d", & voti[i][j] ) ;
        }
    }

    for (i=0; i<K; i++) tot[i]=0 ;
```

```
40      for (j=0; j<N; j++) totg[j]=0 ;

        for (i=0; i<K; i++)
        {
            /* già fatto tot[i] = 0 ; */
45          for (j=0; j<N; j++)
            {
                tot[i] = tot[i] + voti[i][j] ;
                totg[j] = totg[j] + voti[i][j] ;
            }
50      }

        max = tot[0] ;
        posmax = 0 ;
        for (i=1; i<K; i++)
55      {
            if (tot[i]>max)
            {
                max = tot[i];
                posmax = i ;
60          }
        }
```

```
    printf("Il vincitore e' il candidato numero %d\n", posmax+1);

    min = totg[0] ;
    posmin = 0 ;
    for (i=1; i<N; i++)
    {
        if (totg[i]<min)
        {
            min = totg[i];
            posmin = i ;
        }
    }

    printf("Il giudice piu' severo e' il numero %d\n", posmin+1);
    return(0);
}
```

7.2 Statistiche testo

Si scriva un programma in C che acquisisca da tastiera un testo libero, composto da più righe (max 1000) di un numero di caratteri non superiore a 100 ciascuna. L'inserimento termina quando l'utente inserirà una riga uguale a `FINE`.

Al termine dell'acquisizione del testo, il programma dovrà stampare le seguenti statistiche:

1. il numero totale di righe inserite [1];
2. il numero totale di caratteri inseriti;
3. il numero totale di caratteri *alfanumerici* inseriti;
4. il numero totale di *parole* inserite.

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

int main(void)
{
    const int MAX = 1000 ;
    const int LUN = 100 ;

    char testo[MAX][LUN+1] ;
    int N ; /* righe inserite */
    int ncar, nalfa, npar ;
    int end ;

    char riga[300] ;
    int i,j ;
```

```
N = 0 ;
end = 0 ;
do
{
    printf("Testo: ") ;
    gets(riga) ;

    if ( strlen(riga) > LUN )
        printf("Errore: riga troppo lunga (max %d caratteri)\n", LUN) ;
    else if ( strcmp( riga, "FINE" )==0 )
        end = 1 ;
    else
    {
        strcpy( testo[N], riga ) ;
        N++ ;
    }
}
while (end==0 && N<MAX) ;
```

```
printf("L'utente ha inserito %d righe\n", N) ;

ncar = 0 ;
for (i=0; i<N; i++)
    ncar = ncar + strlen( testo[i] ) ;

printf("L'utente ha inserito %d caratteri\n", ncar) ;

nalfa = 0 ;
for (i=0; i<N; i++)
{
    for (j=0;  j<strlen(testo[i])  ; j++)
    {
        if ( isalnum( testo[i][j] ) )
            nalfa++ ;
    }
}

printf("L'utente ha inserito %d caratteri alfanumerici\n", nalfa) ;
```

```
npar = 0 ;
for (i=0; i<N; i++)
{
    for (j=0;  j < strlen(testo[i]) ;     j++)
    {
        /* verifico se [i][j] è il carattere
           iniziale di una parola */
        if ( isalpha(testo[i][j]) &&
                ((j==0)|| !isalpha(testo[i][j-1])) )
        {
            npar++ ;
        }
    }
}

printf("L'utente␣ha␣inserito␣%d␣parole\n", npar) ;
return(0);
}
```

7.4 Gestione magazzino

Un'azienda deve tenere traccia dei beni presenti in un magazzino. L'utente inserisce da tastiera dei "comandi" nel seguente formato:

```
bene EU quantità
```

dove:

- `bene` è il nome di un bene;
- `EU` è la lettera `'E'` per entrata, `'U'` per uscita;
- `quantità` è la quantità di bene entrata o uscita.

L'utente termina il caricamento inserendo un comando pari a `FINE`. In tal caso il programma deve stampare le quantità di beni presenti a magazzino.

Esempio:

```
viti E 10
dadi E 50
viti U 5
viti E 3
FINE
```

N.B. Si assuma che i nomi dei prodotti siano privi di spazi e contengano al più 29 caratteri; si assuma che possano essere presenti al più 100 prodotti diversi.

Soluzione

```c
/* PROGRAMMAZIONE IN C */

/* File: magazzino.c */
/* Soluzione proposta esercizio "Gestione magazzino" */

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

#define MAX 100
#define LUN 30


int main(void)
{
    char prodotti[MAX][LUN] ;
    char prod[LUN] ;
    int quantita[MAX] ;
    int qta ;
    char dir ;
    int i ;
    int trovato ;
    int N ; /* dimensione dei vettori prodotti[] e quantita[] */

    N = 0 ;
```

```
        do
        {
            /* acquisisci un comando dall'utente */

30          /* NOTA: non si può usare il costrutto
                scanf("%s %c %d", prod, &dir, &qta) ;
               in quanto non funziona per l'ultima riga (FINE) */

            printf("Comando: ") ;
35          scanf("%s", prod) ;

            if ( strcmp(prod, "FINE") != 0 )
            {
                scanf(" %c %d", &dir, &qta) ;
40
                if ( dir=='E' ) /* entrata */
                {
                    /* trova la posizione del prodotto nel vettore prodotti[] */
                    trovato = -1 ;
45                  for (i=0; i<N; i++)
                    {
                        if ( strcmp(prodotti[i], prod) == 0 )
                            trovato = i ;
                    }
```

```
            if ( trovato != -1 ) /* prodotto esiste già */
            {
                /* incrementa la posizione corrispondente del vettore
                   quantita[] */
55              quantita[trovato] = quantita[trovato] + qta ;
            }
            else /* prodotto nuovo */
            {
                /* aggiungi il prodotto al magazzino in posizione nuova */
60              strcpy(prodotti[N], prod) ;
                quantita[N] = qta ;

                N++ ;
            }
65      }
```

```
        else /* uscita */
        {
            /* trova la posizione del prodotto nel vettore prodotti[] */
            trovato = -1 ;
70          for (i=0; i<N; i++)
            {
                if ( strcmp(prodotti[i], prod) == 0 )
                    trovato = i ;
            }
7s
            if ( trovato == -1 )
            {
                printf("Prodotto␣%s␣non␣trovato␣in␣magazzino\n", prod);
            }
80          else
            {
```

```
                        /* decrementa la posizione corrispondente del vettore
                           quantita[] */
                        quantita[trovato] = quantita[trovato] - qta ;
                    }
                }
            }

        }
        while ( strcmp(prod, "FINE") != 0 ) ;

        for (i=0; i<N; i++)
        {
            printf("%s %d\n", prodotti[i], quantita[i]) ;
        }

    return(0);
}
```

7.3 Rubrica telefonica

Si realizzi un programma in linguaggio C in grado di gestire una rubrica di nomi e numeri
telefonici. La rubrica deve contenere fino a 100 voci diverse. Ciascuna voce è composta da un nome (max 40 caratteri) e da un numero di telefono (max 20 caratteri).

Il programma deve fornire all'utente un menù di scelta, con le seguenti voci:

1) Aggiungi nuova voce in rubrica
2) Ricerca esatta per nome
3) Ricerca approssimata per nome
4) Stampa completa rubrica
0) Esci dal programma

Una volta che l'utente ha scelto l'operazione desiderata (1-4), il programma acquisirà i dati necessari dall'utente ed eseguirà il comando. Nota: nella rubrica non possono esistere due voci con lo stesso nome.

```c
/* PROGRAMMAZIONE IN C */

/* File: rubrica.c */
/* Soluzione proposta esercizio "Rubrica telefonica" */

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

int main(void)
{
    const int MAX = 100 ; /* numero max di voci */
    const int LUNN = 40 ; /* lunghezza del nome */
    const int LUNT = 20 ; /* lunghezza n. telefono */

    char nome[MAX][LUNN+1] ;
    char tel[MAX][LUNT+1] ;

    int N ; /* numero di voci memorizzate */

    int comando ; /* comando dell'utente 0-4 */

    char riga[200] ;
    char sn[LUNN+1] ;
    char st[LUNT+1] ;
    int i, duplicato, trovato, pos ;

    /* INIZIALIZZAZIONI */

    N = 0 ;
```

```
do
{
    /* STAMPA DEL MENU */
    puts("1) Aggiungi nuova voce in rubrica") ;
    puts("2) Ricerca esatta per nome") ;
    puts("3) Ricerca approssimata per nome") ;
    puts("4) Stampa completa rubrica") ;
    puts("0) Esci dal programma") ;

    /* LETTURA DEL COMANDO */
    printf("Inserisci il comando: ") ;
    gets(riga) ;
```

sscanf(riga, ”%d”, &comando);

```
    /* ESECUZIONE DEL COMANDO */
    switch ( comando )
    {
    case 1:
        /* Acquisisci i dati */
        printf("Inserisci il nome da aggiungere: ") ;
        gets(sn) ;
        printf("Inserisci il numero di telefono corrispondente: ") ;
        gets(st) ;

        /* Verifica se i dati sono validi */
        if ( N == MAX )
        {
            puts("ERRORE: rubrica piena") ;
        }
```

else {

```
            duplicato = 0 ;
            for ( i = 0 ; i < N ; i++ )
65              if ( strcmp(sn, nome[i]) == 0 )
                    duplicato = 1 ;

            if ( duplicato == 1 )
            {
70              puts("ERRORE: nome duplicato") ;

            } else {

            /* Aggiungi il nome in rubrica */
75          strcpy( nome[N], sn ) ;
            strcpy( tel[N], st ) ;
            N++ ;  }}

            break ;
80
        case 2: /* ricerca esatta */
            printf("Inserisci il nome da ricercare: ") ;
            gets(sn) ;

85          trovato = 0 ;
            for ( i = 0 ; i < N && trovato == 0 ; i++ )
            {
                if ( strcmp( sn, nome[i] ) == 0 )
                {
90                  trovato = 1 ;
                    pos = i ;
                }
            }
```

```
95              if ( trovato == 1 )
                {
                    printf("Il telefono di %s e': %s\n",
                           sn, tel[pos] ) ;
                }
100             else
                {
                    printf("Nessun %s e' presente in rubrica\n", sn) ;
                }

105             break ;

            case 3: /* ricerca approssimata */
                printf("Inserisci una parte del nome da ricercare: ") ;
                gets(sn) ;
110
                trovato = 0 ;
                for ( i = 0 ; i < N ; i++ )
                {
                    if ( strstr( nome[i], sn ) != NULL )
115                 {
```

```
                    printf("%s: %s\n", nome[i], tel[i]) ;
                    trovato = 1 ;
                }
            }

            if (trovato==0)
                printf("Non trovato...\n") ;
            break ;

        case 4:
            printf("CONTENUTO DELLA RUBRICA (%d VOCI)\n", N) ;

            for ( i = 0 ; i < N ; i++ )
                printf("%s: %s\n", nome[i], tel[i] ) ;
            break ;

        case 0:
            puts("Arrivederci") ;
            break ;

        default:
            printf("ERRORE NEL PROGRAMMA (comando=%d)\n", comando) ;
        }

    }
    while ( comando != 0 ) ;

    return(0);
}
```