ESERCIZIO SULLE CODE

Sia data una coda statica che memorizza elementi costituiti da:

- provenienza stringa
- destinazione stringa
- priorita' intero

Si scrivano le seguenti procedure:

1.inserimento di un elemento in coda

2.estrazione e stampa dell'elemento estratto

3.stampa del contenuto della coda

4.stampa degli elementi della coda caratterizzati da una data "provenienza"

5.estrazione di tutti gli elementi e stampa contestuale di quelli caratterizzati da una data "priorita'"

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

typedef struct {
  char  provenienza [40];
  char  destinazione [40];
  int   pri;
} t_elemento;

#define QUEUE_SIZE    6

typedef struct {
  int  head, tail, n;
  t_elemento data [QUEUE_SIZE];
} t_queue;

t_queue init_queue (void)
{
  t_queue q;
  q.head = q.tail = q.n = 0;
  return q;
}

int is_queue_full (t_queue q)
{
  return q.n == QUEUE_SIZE;
}

int is_queue_empty (t_queue q)
{
  return q.n == 0;
}
```

```
void in (t_queue * q, t_elemento el)
{
  q->data [q->tail] = el;
  q->tail = (q->tail + 1) % QUEUE_SIZE;
  q->n++;
}

t_elemento out (t_queue * q)
{
  t_elemento e;
  e = q->data [q->head];
  q->head = (q->head + 1) % QUEUE_SIZE;
  q->n--;
  return e;
}

void dump_queue (t_queue q)
{
  int i, index;
  t_elemento e;
  index = q.head;
  for (i = 0; i < q.n; i++) {
        e = q.data [index];
        printf ("Prov. %s\n", e.provenienza);
        printf ("Dest. %s\n", e.destinazione);
        printf ("Pri.  %d\n", e.pri);
        printf ("--------\n");
        index = (index + 1) % QUEUE_SIZE;
  }
}
```

```
void dump_on_provenienza (t_queue q,
                          char * prov)
{
  int i, index;
  t_elemento e;
  index = q.head;
  for (i = 0; i < q.n; i++) {
        e = q.data [index];
        if (strcmp (e.provenienza, prov) == 0) {
              printf ("Prov. %s\n",
                       e.provenienza);
              printf ("Dest. %s\n",
                       e.destinazione);
              printf ("Pri.  %d\n", e.pri);
              printf ("--------\n");
        }
        index = (index + 1) % QUEUE_SIZE;
  }
}

void extract_on_pri (t_queue * q, int pri)
{
  t_elemento e;
  while (is_queue_empty (*q) == 0) {
        e = out (q);
        if (e.pri == pri) {
              printf ("Prov. %s\n",
                       e.provenienza);
              printf ("Dest. %s\n",
                       e.destinazione);
              printf ("Pri.  %d\n", e.pri);
              printf ("--------\n");
        }
  }
}
```

```
int main (int argc, char * argv[])
{
  t_queue myqueue;
  int choice;

  myqueue = init_queue ();
  do {
    printf ("0. Exit\n");
    printf ("1. Inserisci elemento\n");
    printf ("2. Estrai elemento\n");
    printf ("3. Stampa coda\n");
    printf ("4. Stampa in base a provenienza\n");
    printf ("5. Svuota e stampa in base a pr..\n");
    scanf ("%d", &choice);
    switch (choice) {
      case 1:
      {
        t_elemento elem;
        printf ("Inserisci la provenienza :");
        scanf ("%s", elem.provenienza);
        printf ("Inserisci la destinazione :");
        scanf ("%s", elem.destinazione);
        printf ("Inserisci la priorita' :");
        scanf ("%d", &elem.pri);
        if (is_queue_full (myqueue) == 0)
          in (&myqueue, elem);
        else
          printf ("Coda piena\n");
      }
      break;
...
```

```
...
    case 2:
    {
      t_elemento elem;
      if (is_queue_empty (myqueue) == 0) {
        elem = out (&myqueue);
        printf (“Elemento = (%s,%s,%d)\n",
                  elem.provenienza,
                  elem.destinazione,
                  elem.pri);
      }
      else
        printf ("Coda vuota, out fallito\n");
      break;
    }
    case 3:
      dump_queue (myqueue);
      break;
    case 4:
    {
      char prov [40];
      printf ("Inserisci la provenienza:");
      scanf ("%s", prov);
      dump_on_provenienza (myqueue, prov);
      break;
    }
    case 5:
    {
      int pri;
      printf ("Inserisci la priorita':");
      scanf ("%d", &pri);
      extract_on_pri (&myqueue, pri);
      break;
    }
   }
  } while (choice != 0);
}
```