UNIVERSITA` DI BERGAMO

ESAME DI

INFORMATICA 12 CFU – Modulo di Programmazione (ING. INFORMATICA)

Prof. G. PSAILA

APPELLO DEL 18/06/2024

Per consegnare, si svolgano entrambi gli esercizi.
Durata: 90 minuti.
Punteggio complessivo: 16 punti. Sufficienza: 9 punti.

Esercizio (10 punti)
Si consideri un programma per gestire le prenotazioni di una catena di alberghi.
Una prenotazione è descritta da un tipo strutturato denominato `PRENOTAZIONE`, i cui campi sono un codice di 18 caratteri (che identifica univocamente la prenotazione), la data di emissione (stringa nel formato internazionale `aaaa-mm-gg`), la data di inizio del periodo prenotato (stringa di 10 caratteri), la data di fine del periodo prenotato (stringa di 10 caratteri), il codice dell'albergo (stringa di 10 caratteri), il numero della camera (numero intero), la categoria della camera (numero intero), il prezzo della prenotazione (numero in virgola mobile) e lo stato della prenotazione (numero intero, dove 1 indica che la prenotazione è attiva, 2 indica che il soggiorno è avvenuto, 3 indica che la prenotazione è stata annullata).
Si definisca quindi la struttura dati per una lista dinamica dove il campo informativo del nodo è a sua volta basato sul tipo `PRENOTAZIONE`.
Si scriva la funzione denominata `PrenotazioniAlbergoMaxDurataMedia`, che riceve come parametri una lista di prenotazioni denominata `listaIn`, un vettore di caratteri denominato `dataInizio` (che descrive una data in formato internazionale), un vettore di stringhe (vettore di puntatori a carattere) denominato `alberghi` ed un numero intero denominato `size`, che indica il numero di elementi nel vettore `alberghi`. Il vettore `alberghi` contiene un elenco di codici di albergo.
La funzione restituisce una nuova lista di prenotazioni, così ottenuta: (1) per ciascun albergo riportato nel vettore `alberghi`, la funzione calcola la durata media dei soggiorni (descritti dalle prenotazioni per quell'albergo) che iniziano nella data `dataInizio`; (2) trovato l'albergo con il massimo valore per la durata media dei soggiorni di interesse, la funzione crea una nuova lista (gestita internamente con la variabile `listaOut`) che contiene una copia di tutte le prenotazioni per quell'albergo e relative a soggiorni che iniziano in data `dataInizio` (cioè, i soggiorni di interesse usati per calcolare la durata media al punto 1). La funzione restituisce il valore di `listaOut`, oppure, in caso di errore, la funzione restituisce il valore `NULL`.
N.B. Si risolva il problema in modo da fare esattamente DUE scansioni della lista puntata dal parametro `listaIn`, lavorando direttamente su di essa e senza creare copie temporanee (sia totali che parziali).
N.B. Si eviti la duplicazione di parti del codice del programma.

Domanda Teoria (6 punti)
Si consideri un sistema in cui per gli indirizzi di memoria vengono usati 24 bit e la memoria viene gestita con il sistema della paginazione con pagine da 1Kbyte e indirizzi logici. Si consideri il seguente indirizzo logico *l*=000000000100100000001101.
Se nella tabella delle pagine abbiamo le corrispondenze *pl*→*pf* (in base 10) 29→18, 32→127, 18→19, 20→13, qual è l'indirizzo fisico *f* su 24 bit corrispondente all'indirizzo logico *l* ?

```
/*
Soluzione appello del 18/06/2024
*/

#include <iostream>
#include <cstring>

using namespace std;

/*
 *
 Struttura dati
 */

// funzione presente nel programma che calcola la distanza,
// in giorni, tra due date.
// Durante l'esame è stato detto di usarla, senza preoccuparsi
// di implemnetarla
int get_giorni(char data_inizio[], char data_fine[])
{
     return 0;
}

struct PRENOTAZIONE
{
     char codice [18+1];
     char data_emissione [10+1];
     char data_inizio [10+1];
     char data_fine [10+1];
     char albergo [10+1];
     int camera;
     int categoria_camera;
     float prezzo;
     int stato;
};

struct NODO
{
     PRENOTAZIONE dato;
     NODO *next;
};

int ins_testa( NODO *&head, PRENOTAZIONE dato )
{
     NODO *t;

     t = new NODO;
     if( t == NULL )
     {
          cout << "Errore Allocazione";
          return 1;
     }

     t->dato = dato;
     t->next = head;
```

```
     head = t;

     return 0;
}

int cerca_albergo(char albergo[], char *alberghi[], int size)
{
     for(int i=0; i < size; i++)
     {
          if( strcmp(albergo, alberghi[i])==0 )
               return i;
     }

     return -1;   // albergo non trovato
}

// con una sola scansione, calcola i contatori e i sommatori
void Calcola(NODO *listaIn,char *alberghi[],
                      int size, int contatori[],
                           int sommatori[], char dataInizio[])
{
     NODO *p;

     int i;
     int pos_albergo;

     //inizializzazione dei contatori
     for(i=0; i < size; i++)
     {
          contatori[i]=0;
          sommatori[i]=0;
     }

     // Una sola scansione per calcolare
     // i contatori e i sommatori
     p=listaIn;
     while(p != NULL)
     {
          pos_albergo = cerca_albergo(p->dato.albergo,
                                      alberghi, size);
          if( pos_albergo >= 0)
          {
               if( strcmp(p->dato.data_inizio, dataInizio)==0 )
               {
                    contatori[pos_albergo]++;
                    sommatori[pos_albergo] +=
                       get_giorni(p->dato.data_inizio,
                                  p->dato.data_fine);
               }
          }
          p = p->next;
     }
}
```

```
int CalcolaMax(int contatori[], int sommatori[], int size)
{
     int max_pos=-1;
     float max = -1;
     float media;

     for(int i=0; i < size; i++)
     {
          if( contatori[i] > 0 )
          {
               media = sommatori[i] / (float)contatori[i];
               if(media > max)
               {
                    max_pos = i;
                    max = media;
               }
          }
     }

     return max_pos;
}

NODO *CopiaPrenotazioni(NODO *listaIn,
                        char albergo[], char dataInizio[])
{
     NODO *listaOut = NULL;

     NODO *p;

     p = listaIn;
     while(p != NULL)
     {
          if( strcmp(p->dato.albergo, albergo)==0 &&
              strcmp(p->dato.data_inizio, dataInizio)==0 )
          {
               if( ins_testa(listaOut, p->dato))
                    return NULL;
          }
          p = p->next;
     }
     return listaOut;
}


NODO *PrenotazioniAlbergoMaxDurataMedia(NODO *listaIn,
             char dataInizio[],char *alberghi[], int size)
{
     if(size < 1)
          return NULL;

     int contatori[size];
     int sommatori[size];

     Calcola(listaIn, alberghi, size,
              contatori, sommatori, dataInizio);
```

```
    int max_pos = CalcolaMax(contatori, sommatori, size);
    if( max_pos < 0)
        return NULL;
    NODO *listaOut = CopiaPrenotazioni(listaIn,
                        alberghi[max_pos], dataInizio);

    return listaOut;
}
```