UNIVERSITA` DI BERGAMO

ESAME DI

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

Prof. G. PSAILA

APPELLO DEL 02/07/2024

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 `PrenotazioniAlbergoMaxDueMesi`, che riceve come parametri una lista di prenotazioni denominata `listaIn`, 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 conta il numero di prenotazioni il cui soggiorno termina nello stesso anno in cui è iniziato, ma nel mese successivo (per esempio, se la prenotazione è iniziata nel mese 09, deve terminare nel mese 10 dello stesso anno); (2) trovato l'albergo con il massimo di prenotazioni con le caratteristiche cercate, la funzione crea una nuova lista (gestita internamente con la variabile `listaOut`) che contiene una copia di tutte le prenotazioni cercate (cioè quelle che finiscono nel mese successivo al mese di inizio, ma nello stesso anno) per quell'albergo. 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 l'encoding dei caratteri in UTF-8. Dati i due byte DE 95 (in base 16) che descrivono un carattere in UTF-8, si estragga il codice binario e si riporti il corrispondente valore in base 10 del carattere corrispondente.

```
/*
Soluzione appello del 02/07/2024
*/

#include <iostream>
#include <cstring>

using namespace std;

/*
 *
 Struttura dati
 */
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 estrai_valore(char data[], int ini, int len)
{
     char buffer[len+1];
     int i;
```

```
     for(i=0; i < len; i++)
          buffer[i]=data[ini+i];
     buffer[i]='\0';

     return atoi(buffer);
}

int verifica_date(char inizio[], char fine[])
{
     int anno1, anno2, mese1, mese2;

     anno1 = estrai_valore(inizio, 0, 4);
     anno2 = estrai_valore(fine, 0, 4);
     mese1 = estrai_valore(inizio, 4, 2);
     mese2 = estrai_valore(fine, 4, 2);

     if( anno1 == anno2 && mese1 + 1 == mese2)
          return 1;

     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
}

void CalcolaContatori(NODO *listaIn,char *alberghi[],
                      int size, int contatori[])
{
     NODO *p;

     int i;
     int pos_albergo;

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

     // Una sola scansione per calcolare i contatori
     p=listaIn;
     while(p != NULL)
     {
          pos_albergo = cerca_albergo(p->dato.albergo,
                                      alberghi, size);
          if( pos_albergo >= 0)
          {
               if( verifica_date(p->dato.data_inizio,
```

```
                                        p->dato.data_fine) )
                        contatori[pos_albergo]++;
                }
                p = p->next;
        }
}

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

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

        return max_pos;
}

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

        NODO *p;

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

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

        int contatori[size];
```

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

    return listaOut;
}
```