UFU

Nome: Antonio Carlos Neto

Curso: Ciência da Computação

Disciplina: GBC024

MAIN.C:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "list.h"

#include "stack.h"

char plate[15], temp[15] ;

int main (int argc,char \*argv[])

{

char ad ;

int i,s, lane = -1, min, lc ;

Lista lista[LANES];

for(i=0; i<LANES; i++)

{

list\_init(&lista[i]);

}

while (1)

{

for ( i = 0 ; i < LANES ; i++ )

{

printf( "lane %d: ", i ) ;

q\_display ( &lista[i] ) ;

}

printf( "\nArrival/Departure/Quit? ( A/D/Q ): " ) ;

ad = getch( ) ;

if ( toupper ( ad ) == 'Q' ) exit ( 1 ) ;

printf ( "\nEnter license plate num:" ) ;

gets ( plate ) ;

ad = toupper ( ad ) ;

if ( ad == 'A' ) /\* arrival of car \*/

{

lane = -1 ; /\* assume no lane is available \*/

min = CAPACITY ;

for ( i = 0 ; i < LANES ; i++ )

{

s = lista[i].size;

if ( s < min )

{

min = s ;

lane = i ;

}

}

if ( lane == -1 )

printf ( "\nNo room available" ) ;

else

{

add\_inicio ( &lista[lane], plate ) ;

printf ( "\npark car at lane %d slot %d\n", lane, s ) ;

}

}

else

{

if ( ad == 'D' ) /\* departure of car \*/

{

for ( i = 0 ; i < LANES ; ++i )

{

s = search ( &lista[i], plate ) ;

if ( s != -1 )

{

lane = i ;

break ;

}

}

if ( i == LANES )

printf ( "\nno such car!!\n" ) ;

else

{

printf ( "\ncar found at lane %d slot %d\n", lane, s ) ;

excluir (&lista[ lane ], s ) ;

}

}

else if ( ad == 'Q' )

exit ( 1 ) ;

}

}

}

LIST.C:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "list.h"

#include "stack.h"

char temp[15];

void q\_display(Lista \*lista)

{

int i;

Lista \*q = lista;

for(i=0 ; i<lista->size ; i++)

{

printf("%s", q->head->plate);

q->head = q->head->next;

}

}

int search(Lista \*q, char \*p)

{

int i, s = -1, c = 0 ;

Lista \*lista = q;

for(i=0 ; i<lista->size ; i++)

{

if (strcmp (p, lista->head->plate) == 0)

{

s = c ;

break ;

}

else

{

lista->head = lista->head->next;

c++ ;

}

}

return(s);

}

void add\_inicio(Lista \*lista, char \*p)

{

node \*q;

q = (node\*) malloc(sizeof(node));

strcpy(q->plate,p);

if(lista->size == 0)

{

lista->head = q;

q->next = NULL;

}else

{

lista->tail->next = q;

q->next = NULL;

}

lista->tail = q;

lista->size++;

}

void add\_fim(Lista \*lista, char \*p)

{

node \*q;

node \*r;

r = lista->head;

q = (node\*) malloc(sizeof(node));

strcpy(q->plate,p);

if(lista->size == 0)

{

lista->head = q;

q->next = NULL;

lista->tail = q;

}

else

{

q->next = lista->head;

lista->head = q;

}

lista->size++;

}

void list\_init(Lista \*list)

{

list->size = 0;

list->head = NULL;

list->tail = NULL;

}

char\* excluir(Lista \*lista, int n)

{

int i,j;

node \*q;

node aux;

Lista stack;

list\_init(&stack);

if(lista->size == 0)

{

printf("\nLista Vazia");

}else

{

if(n==0)

{

strcpy(temp, lista->head->plate);

q = lista->head;

lista->head = lista->head->next;

free(q);

lista->size--;

return temp;

}

for(i=0; i<n ; i++)

{

j = push(&stack, lista->head->plate);

printf("Driving out number's plate %s\n",lista->head->plate);

lista->size--;

lista->head = q->next;

free(q);

}

q = lista->head;

lista->head = q -> next;

free(q);

lista->size--;

for(i=0; i<n ; i++)

{

j = pop(&stack,&aux);

strcpy(temp, aux.plate);

add\_fim(lista, temp);

}

}

}

LIST.H :

#ifndef LIST\_H\_INCLUDED

#define LIST\_H\_INCLUDED

#define LANES 5

#define CAPACITY 10

typedef struct node\_

{

char plate[15];

struct node \*next;

}node;

typedef struct Lista\_

{

node \*head;

node \*tail;

int size;

}Lista;

void q\_display(Lista \*lista);

int search(Lista \*q, char \*p);

void add\_inicio(Lista \*lista, char \*p);

void add\_fim(Lista \*lista, char \*p);

void list\_init(Lista \*list);

char\* excluir(Lista \*lista, int n);

#endif // LIST\_H\_INCLUDED

STACK. C:

#include <stdio.h>

#include <stdlib.h>

#include "list.h"

#include "stack.h"

char temp[15];

int push(Lista \*lista, char placa)

{

node \*q;

q = (node\*)malloc(sizeof(node));

if(q == NULL) return -1;

strcpy( q->plate, placa ) ;

q->next = lista->head;

lista->head = q;

lista->size++;

return 0;

}

int pop(Lista \*lista, node \*s)

{

node \*excluido;

if (lista->size == 0)

{

printf("\nERRO, PILHA VAZIA");

return -1;

}

else

{

excluido = lista->head;

strcpy(node->plate, excluido -> plate);

strcpy(temp, lista->plate);

lista->head = excluido->next;

free(excluido);

return 0;

}

}

STACK.H :

#ifndef STACK\_H\_INCLUDED

#define STACK\_H\_INCLUDED

#include <stdlib.h>

#include "list.h"

int push(Lista \*lista, char placa);

int pop(Lista \*lista, node \*s);

#endif // STACK\_H\_INCLUDED