typedef enum

{

false,

true

} Bool;

/\*Defintion d'une liste\*/

typedef struct ListeElement

{

int value;

struct ListeElement \*next;

} ListeElement, \*List;

#include <stdio.h>

#include <stdlib.h>

#include "list.h"

List new\_list(void)

{

return NULL;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Bool is\_empty\_list(List li)

{

if(li == NULL)

{

return true;

}

return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int list\_length(List li)

{

int size = 0;

if(is\_empty\_list(li))

return size;

while(li != NULL)

{

++size;

li = li->next;

}

return size;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void print\_list(List li)

{

if(is\_empty\_list(li))

{

printf("\nLa liste est vide. Donc rien a afficher");

return;

}

while(li != NULL)

{

printf("[%d]", li->value);

li = li->next;

}

printf("\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int first\_element(List li)

{

if(is\_empty\_list(li))

{

return 0;

}

return li->value;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

List push\_back\_list(List li,int x)

{

ListeElement \*element;

element = malloc(sizeof(\*element));

if(element == NULL)

{

fprintf(stderr, "Erreur d'allocation");

exit(EXIT\_FAILURE);

}

element->value = x;

element->next = NULL;

if(is\_empty\_list(li))

return element;

ListeElement \*temp;

temp = li;

while(temp->next != NULL)

temp = temp->next;

temp->next = element;

return li;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

List push\_from\_list(List li,int x)

{

ListeElement \*element;

element = malloc(sizeof(\*element));

if(element == NULL)

{

printf("Probleme d'allocation");

exit(EXIT\_FAILURE);

}

element->value = x;

if(is\_empty\_list(li))

element->next = NULL;

else

element->next = li;

return element;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

List pop\_back\_list(List li)

{

if(is\_empty\_list(li))

return new\_list();

if(li->next == NULL)

{

free(li);

li=NULL;

return NULL;

}

ListeElement \*temp = li;

ListeElement \*before = li;

while(temp->next != NULL)

{

before = temp;

temp = temp->next;

}

before->next = NULL;

free(temp);

temp = NULL;

return li;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

List pop\_from\_list(List li)

{

if(is\_empty\_list(li))

return new\_list();

ListeElement \*element;

element = malloc(sizeof(\*element));

if(element == NULL)

{

fprintf(stderr, "Erreur d'allocation");

exit(EXIT\_FAILURE);

}

element = li->next;

free(li);

li = NULL;

return element;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int last\_element(List li)

{

int value;

if(is\_empty\_list(li))

return 0;

while(li->next != NULL)

{

li = li->next;

}

return li->value;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

List clear\_list(List li)

{

if(is\_empty\_list(li))

return NULL;

while(li != NULL)

{

li = pop\_back\_list(li);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*int search\_element(List li, int x)

{

if(is\_empty\_list(li))

printf("\nIl n'y a rien dans la liste");

while(li->next != NULL)

{

if(x = li->value)

printf("Element trouver");

else

li= li->next;

}

}\*/

List myList;

myList = new\_list();

if(is\_empty\_list(myList))

{

printf("Vide");

}

else

{

printf("Pas vide");

}

printf("\nLa taille de la liste est:%d\n", list\_length(myList));

myList = push\_back\_list(myList, 36);

myList = push\_back\_list(myList, 38);

myList = push\_back\_list(myList, 32);

myList = push\_back\_list(myList, 30);

myList = push\_back\_list(myList, 33);

myList = push\_back\_list(myList, 33);

print\_list(myList);

search\_element(myList, 666);