**מעבדה 12 – מת"מ -   
מגישים:  
אורי מלכא – 314862996  
אלן ציפין - 313206062  
  
שאלה 1 מיין:  
  
שאלה 1 מימוש:**

#include "Lab12Matam.h"

//compare\_func

void free\_char(char \*c) {

free(c);

}

BOOL cmp\_char(char\* a, char\* b) {

if (\*a == \*b)

return TRUE;

return FALSE;

}

void prnt\_char(char \*c) {

printf("%c", \*c);

}

int main()

{

char ch, \* temp;

int n, i;

BOOL res;

PNode head = NULL, tail = NULL;

printf("Enter number of characters ");

scanf("%d", &n);

printf("\nEnter %d characters,separated by enter", n);

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

{

temp = (char\*)malloc(sizeof(char));

if (NULL == temp)

{

printf("\nCan't allocate data memory");

freeAll(&head, free\_char);

return 1;

}

scanf(" %c", temp);

res = insertEntry(&head, &tail, temp, cmp\_char);

if (FALSE == res) /\* Can't allocate node memory or data is already exists\*/

{

printf("\nCan't allocate node memory or data is already exists");

freeAll(&head, free\_char);

return 1;

}

}

printAll(head, prnt\_char);

printf("\nEnter an element for deleting");

scanf(" %c", &ch);

res = deleteNode(&head, &tail, &ch, cmp\_char, free\_char);

if (FALSE == res)

printf("%c don't exists in list\n", ch);

printAll(head, prnt\_char);

freeAll(&head, free\_char);

printf("\nYour list was destroyed\n");

return 0;

}

#include "Lab12Matam.h"

typedef struct ListNode

{

void\* data;

struct ListNode\* next;

} ListNode;

BOOL insertEntry(PNode\* head, PNode\* tail, void\* data, compare\_func func) {

PNode temp, q;

q = \*head;

temp = (PNode)malloc(sizeof(ListNode));

if (temp == NULL)

return FALSE; //in case that memo allocation failed.

temp->data = data;

temp->next = NULL;

if (\*head == NULL) { //empty list

\*head = temp;

\*tail = temp;

return TRUE;

}

while (q != NULL) {

if (func(q->data, temp->data)==TRUE) {

printf("data is already exist");

return FALSE;

}

q = q->next;

}

// if we got here we need to add the node.

(\*tail)->next = temp;

\*tail = temp;

return TRUE;

}

BOOL deleteNode(PNode\* head, PNode\* tail, void\* todel, compare\_func func1, free\_func func2) {

PNode temp, q;

q = \*head;

//case for the first node in the list (delete from the head)

if (func1(q->data, todel)) {

temp = \*head;

\*head = (\*head)->next;

func2(temp->data);

free(temp);

return TRUE;

}

//once we got here we know that the first NODE in the list is not the node that we want to delete.

while (q->next != NULL) {

if (func1(q->next->data, todel)==TRUE) {

temp = q->next; //the node that we want to delete.

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

func2(temp->data);

free(temp);

return TRUE;

}

q = q->next;

}

return FALSE; // we couldn't find the NODE that stores the "todel" provided data.

}

void printAll(PNode head, print\_func func) {

PNode q = head;

while (q != NULL) {

func(q->data);

q = q->next;

}

}

void freeAll(PNode\* head, free\_func func) {

PNode temp;

while (\*head != NULL) {

temp = \*head;

\*head = (\*head)->next;

func(temp->data);

free(temp);

}

}

**שאלה 1 הדר:**

#ifndef \_Lab12MatamImpl

#define \_Lab12MatamImpl

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

typedef enum { FALSE, TRUE } BOOL;

typedef BOOL (\*compare\_func)(void\*, void\*);

typedef void (\*print\_func)(void\*);

typedef void (\*free\_func)(void\*);

typedef struct ListNode\* PNode;

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

/\* public interface: \*/

/\* inserting a new element.\*/

BOOL insertEntry(PNode\* head, PNode\* tail, void\* data,

compare\_func func);

/\* deleting an element,pointered by todel\*/

BOOL deleteNode(PNode\* head, PNode\* tail, void\* todel,

compare\_func func1, free\_func func2);

/\* print the elements in the list \*/

void printAll(PNode head, print\_func (func));

/\* deleting the entire list \*/

void freeAll(PNode\* head, free\_func func);

#endif