**דוח מעבדה 11**

**שמות מגישים:**

סער ויקטור – 312392822

אילון בן סימון – 312162951

**תרגיל 1**

קוד התכנית:  
header :

#ifndef example1

#define example1

#include <stdio.h>

#include <stdlib.h>

#include<conio.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 // !example1

Main:

#include "example1.h"

//Define specific functions

//The function checks if two chars are equal

BOOL cmp\_char(void\*a, void\*b)

{

if (\*(char\*)a == \*(char\*)b)

return TRUE;

return FALSE;

}

//The function free char

void free\_char(void\* ch)

{

free(ch);

}

//The function print char

void prnt\_char(void\* ch)

{

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

}

int main()

{

char ch, \*temp;

int n, i;

BOOL res;

PNode head = NULL, tail = NULL;//Initialization to head and tail

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));//Creates a specific address for each variable that is scan from the user

if (NULL == temp)//Check if the allocation was successful

{

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

freeAll(&head, free\_char);//Free the linked list

return 1;

}

scanf(" %c", temp);

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

if (FALSE == res) /\* if FALSE==res than there is something wrong with malloc \*/

{

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

freeAll(&head, free\_char);

return 1;

}

}

printAll(head, prnt\_char);//Print the linked list

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);//Print the linked list

freeAll(&head, free\_char);//Free the linked list

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

getch();

return 0;

}

Implementation:

#include "example1.h"

//Define general functions and structures

typedef struct ListNode

{

void\* data;

struct ListNode \*next;

} ListNode;

//The function creates a linked list

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

{

PNode temp=\*head;

while ((temp) != NULL)//A loop that checks that the information is not already in the linked list

{

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

return FALSE;

temp = temp->next;

}

temp = (PNode)malloc(sizeof(ListNode));//Create a node for information not in the linked list

if (temp == NULL)//Check if the allocation was successful

{

exit(1);

}

temp->data = data;

temp->next = NULL;

if (\*head == NULL)

{

\*head = temp;

\*tail = temp;

}

else

{

(\*tail)->next = temp;

(\*tail) = temp;

}

return TRUE;

}

//A function that deletes a node from the linked list according to a given data from the user

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

{

PNode temp = \*head;

PNode helper;

if (\*head == NULL)

return FALSE;

if(func1(todel,(\*head)->data)==TRUE)

{

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

func2(temp->data);

free(temp);

}

else

{

while (temp != NULL&&temp->next != NULL)

{

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

{

helper = temp->next;

temp->next = temp->next->next;

func2(helper->data);

free(helper);

return TRUE;

}

temp = temp->next;

}

}

return FALSE;

}

//A function that deletes the entire linked list

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

{

PNode temp;

while (\*head != NULL)

{

temp = \*head;

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

func(temp->data);

free(temp);

}

}

//General function for printing

void printAll(PNode head, print\_func func)

{

PNode temp = head;

while (temp!=NULL)

{

func(temp->data);

temp = temp->next;

}

}

פלט:

