**דוח מעבדה 7**

**שמות מגישים:**

סער ויקטור – 312392822

אילון בן סימון – 312162951

**תרגיל 1**

קוד התכנית:

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

//structures definition

typedef struct Item

{

int num;

struct Item\* next;

}Item;

typedef struct List

{

int count;

Item \*head;

Item \*tail;

}List;

//function declerations

void Error\_Msg(char\*);

void AddAsFirst(Item\*a, List\*L);

void AddAsLast(Item\*, List\*);

void MoveToAnotherList(List\* L, List\* Posit, List\* Negat);

void CreateList(List\*, FILE \*);

void PrintItem(Item\*);

void PrintList(List\*, char\*);

void DeleteList(List\* L);

int main()

{

List L, Posit, Negat;

FILE \*in = fopen("ThreeLists.txt", "rt");

if (in == NULL)

Error\_Msg("input file is wrong");

L.head = NULL;

L.tail = NULL;

L.count = 0;

Posit.head = NULL;

Posit.tail = NULL;

Posit.count = 0;

Negat.head = NULL;

Negat.tail = NULL;

Negat.count = 0;

CreateList(&L, in);

PrintList(&L, "\nMy List:\n");

MoveToAnotherList(&L, &Posit, &Negat);

PrintList(&Posit, "\n\nThe Positive List:\n");

PrintList(&Negat, "\n\nThe Negative List:\n\n");

fclose(in);

DeleteList(&Posit);

DeleteList(&Negat);

getch();

return 0;

}

//the function represents that an error occured

void Error\_Msg(char\* msg)

{

printf("\n%s", msg);

exit(1);

}

//the function creates a linked list

void CreateList(List\* L, FILE \*f)

{

int value;

Item \*temp;

while (fscanf(f, "%d", &value) == 1)

{

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

if (temp == NULL)

{

DeleteList(L);

Error\_Msg("Memmory!");

}

temp->num = value;

temp->next = NULL;

if (L->head == NULL)

L->head = temp;

else

L->tail->next = temp;

L->tail = temp;

L->count++;

}

}

//the function prints one item

void PrintItem(Item\* node)

{

printf("%d--> ", node->num);

}

//the function prints the entire linked list

void PrintList(List\* L, char\* title)

{

Item\* temp = L->head;

printf("%s", title);

while (temp)

{

PrintItem(temp);

temp = temp->next;

}

printf("\nThere are %d items in the list", L->count);

}

//the function adds a node to the head of the list

void AddAsFirst(Item\* ptr, List\* L)

{

ptr->next = L->head;

L->head = ptr;

L->count++;

}

//the function adds a node to the tail of the list

void AddAsLast(Item\* ptr, List\* L)

{

ptr->next = NULL;

if (L->head == NULL)

L->head = ptr;

else

L->tail->next = ptr;

L->tail = ptr;

L->count++;

}

//the function deletes the linked list

void DeleteList(List\* L)

{

Item \*temp;

while (L->head != NULL)

{

temp = L->head;

L->head = L->head->next;

free(temp);

}

}

//the function creates new lists accordingly

void MoveToAnotherList(List\*L, List\*Posit, List\*Negat)

{

Item \*temp;

while (L->head != NULL)

{

temp = L->head;

L->head = L->head->next;

if (temp->num >= 0)

AddAsLast(temp, Posit);

else

AddAsFirst(temp, Negat);

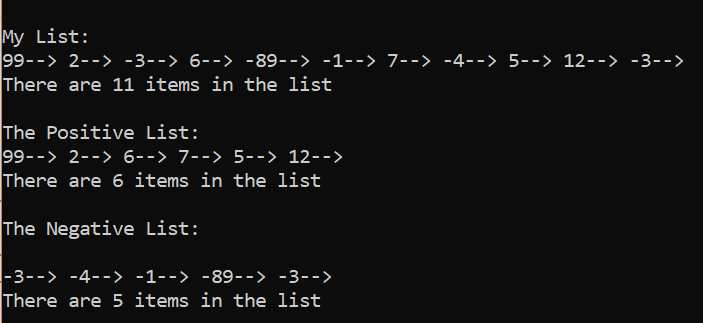
}

L->count = 0;

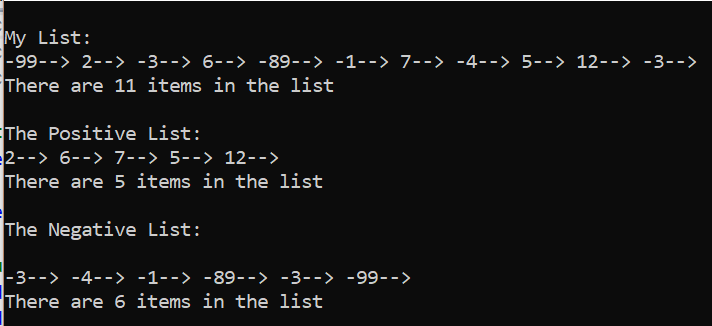
L->tail = NULL;

}

פלט 1:



פלט 2:



**תרגיל 2**

קוד התכנית:

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#define N 5

//structer definition

typedef struct Item

{

int num;

struct Item\* next;

}\*PItem;

//functions declaration

void Error\_Msg(char\*);

void CreateListFromArray(PItem\*, PItem\*, int \*);

void DeleteList(PItem \*);

void ListDisplay(PItem list);

int ListLength(PItem list);

int main()

{

int Arr[N] = { 3,4,1,0,8 };

PItem list = NULL, tail = NULL;

CreateListFromArray(&list, &tail, Arr);

printf("The length of the list is %d members\n", ListLength(list));

printf("\nThe list is:\n");

ListDisplay(list);

DeleteList(&list);

tail = NULL;

getch();

return 0;

}

void Error\_Msg(char\*msg)

{

printf("\n%s", msg);

exit(1);

}

//the function builds a linked list from the array

void CreateListFromArray(PItem\* head, PItem\* tail, int \*Arr)

{

int i;

PItem temp;

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

{

temp = (PItem)malloc(sizeof(struct Item));

if (temp == NULL)

{

DeleteList(head);

Error\_Msg("Memmory!");

}

temp->num = Arr[i];

temp->next = NULL;

if (\*head == NULL)

\*head = temp;

else

(\*tail)->next = temp;

\*tail = temp;

}

}

//the function deletes all the linked list's nodes

void DeleteList(PItem \*head)

{

PItem tmp = \*head;

while (\*head)

{

tmp = \*head;

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

free(tmp);

}

}

//the recursive function returns the number of nodes in the list

int ListLength(PItem list)

{

if (list == NULL)

return 0;

return

ListLength(list->next) + 1;

}

//the recursive function prints the linked list

void ListDisplay(PItem list)

{

if (list != NULL)

{

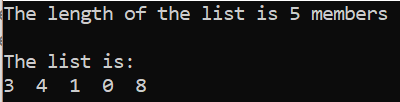
printf("%d ", list->num);

ListDisplay(list->next);

}

}

פלט:



**תרגיל 3**

קוד התכנית:

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

//structure declaration

typedef struct Item

{

int num;

float price;

struct Item\* next;

}\*PItem;

//functions definition

void Error\_Msg(char\*);

void AddNewItem(PItem \*head, PItem \*tail, int place, int a, float b);

void CreateList(PItem \*, PItem \*, FILE \*);

void PrintItem(PItem);

void PrintList(PItem, char\*);

void WriteListToFile(PItem head, FILE \*in);

void DeleteList(PItem \*);

int main()

{

int a, place;

float b;

PItem head = NULL, tail = NULL;

FILE \*in = fopen("input\_price.txt", "rt");

if (in == NULL)

Error\_Msg("input file is wrong");

CreateList(&head, &tail, in);

PrintList(head, "\nThe Old List:\n");

printf("\n\nEnter a number and the price\n");

scanf("%d%f", &a, &b);

printf("\nEnter a place for the new item:");

scanf("%d", &place);

AddNewItem(&head, &tail, place, a, b);

PrintList(head, "\nThe New List:\n");

fclose(in);

in = fopen("input\_price.txt", "wt");

if (in == NULL)

Error\_Msg("input file is wrong");

WriteListToFile(head, in);

DeleteList(&head);

tail = NULL;

fclose(in);

getch();

return 0;

}

//the function represents that an error occured

void Error\_Msg(char\*msg)

{

printf("\n%s", msg);

exit(1);

}

//the function creates a linked list

void CreateList(PItem \*head, PItem \*tail, FILE \*f)

{

int a;

float b;

PItem temp;

while (fscanf(f, "%d %f", &a, &b) == 2)

{

temp = (PItem)malloc(sizeof(struct Item));

if (temp == NULL)

{

DeleteList(head);

Error\_Msg("Memmory!");

}

temp->num = a;

temp->price = b;

temp->next = NULL;

if (\*head == NULL)

\*head = temp;

else

(\*tail)->next = temp;

\*tail = temp;

}

}

//the function prints one item

void PrintItem(PItem node)

{

printf("%d,%.1f-->", node->num, node->price);

}

//the function prints the entire linked list

void PrintList(PItem head, char\* title)

{

printf("%s", title);

while (head)

{

PrintItem(head);

head = head->next;

}

}

//the function deletes the linked list

void DeleteList(PItem \*head)

{

PItem tmp = \*head;

while (\*head)

{

tmp = \*head;

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

free(tmp);

}

}

//the function puts the new item in it's right place

void AddNewItem(PItem \*head, PItem \*tail, int place, int a, float b)

{

int i = 1;

int counter = 0;

PItem ptr = \*head;

PItem temp;

temp = (PItem)malloc(sizeof(struct Item));

if (temp == NULL)

Error\_Msg("Memory!");

temp->num = a;

temp->price = b;

while (ptr != NULL) //the loop counts how many nodes are in the list

{

counter++;

ptr = ptr->next;

}

ptr = \*head;

if (place > counter) //in case the new location is greater than the amount of nodes

{

(\*tail)->next = temp;

\*tail = temp;

(\*tail)->next = NULL;

}

else if (place == 1) //in case the new item needs to be added first

{

temp->next = \*head;

\*head = temp;

}

else //in case the new item needs to be added in the middle of the list

{

while (ptr != NULL)

{

if (i == place-1)

break;

ptr = ptr->next;

i++;

}

temp->next = ptr->next;

ptr->next = temp;

}

}

//the function prints to the file the new linked list

void WriteListToFile(PItem head, FILE \*in)

{

while (head != NULL)

{

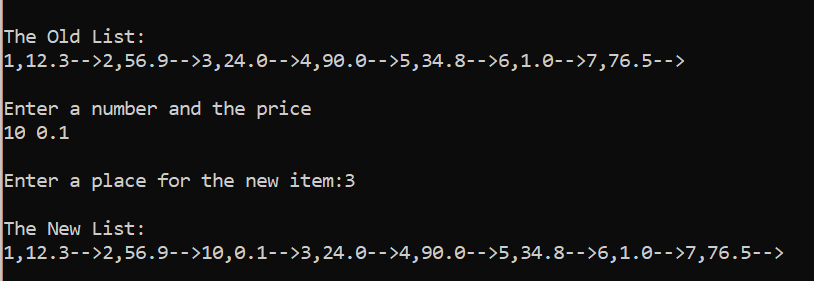
fprintf(in, "%d %.1f ", head->num, head->price);

head = head->next;

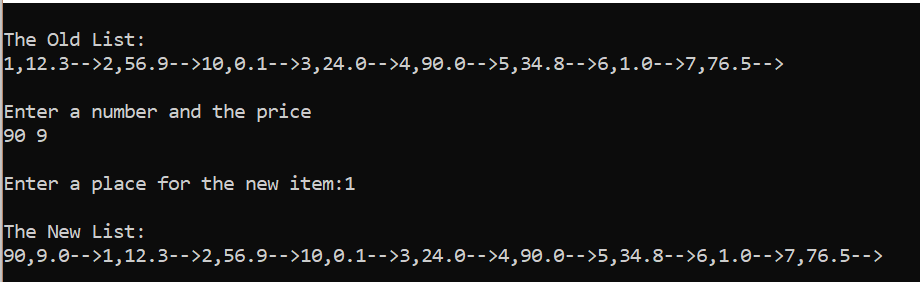
}

}

פלט 1:



פלט 2:



פלט 3:

