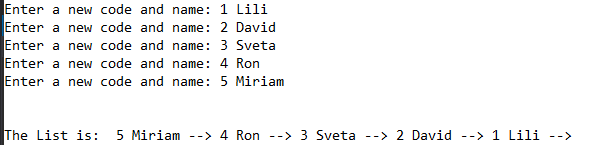
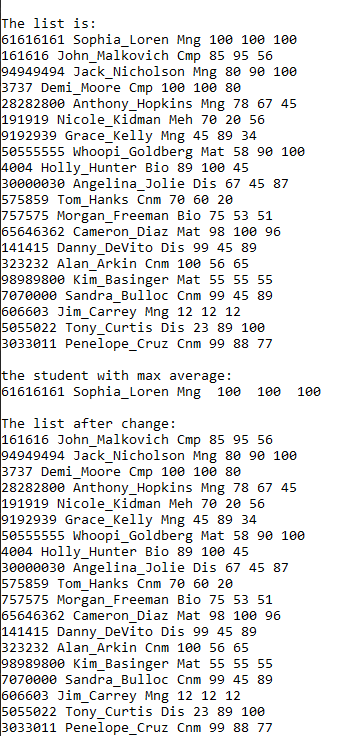
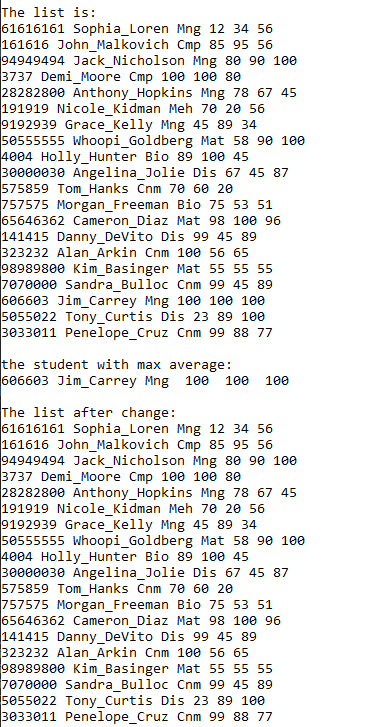
**מעבדה 6 – מת"מ -   
מגישים:  
אורי מלכא – 314862996  
אלן ציפין - 313206062  
  
שאלה 1 קוד:  
שאלה 1 פלט:** **שאלה 2 קוד:  
  
  
  
שאלה 2 פלטים:  
פלט1 מחיקת איבר מראש הרשימה:** **פלט 2 מחיקת איבר לא מראש הרשימה:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N 5

struct Item

{

int code;

char name[11];

struct Item \*next;

};

int main()

{

int i;

struct Item \*Head = NULL, \*temp;

for (i = 1; i <= N; i++) {

temp = (struct Item\*)malloc(sizeof(struct Item)); //making a new Node

if (temp == NULL) {

printf("no memo"); // if memo allocation failed

getchar();

exit(1);

}

printf("Enter a new code and name: ");

scanf("%d %s", &temp->code, temp->name);

temp->next = Head; //the last node in the list, next=NULL (and every other iteration initialized to the next node)

Head = temp;

}

printf("\n\nThe List is: "); //prints the list

while (temp!=NULL) {

printf("%d %s --> ", temp->code, temp->name);

temp = temp->next;

}

while (Head) { //delete the nodes

temp = Head;

Head = Head->next;

free(temp);

}

return 0;

}

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct std

{

char code[11];

char \*name;

char Dep[4];

int marks[3];

float avg;

struct std\* next;

}std;

void Error\_Msg(char\* str);

/\*

Function name:Error\_Msg

Input:char\*

Output:VOID

Function Algorithm:prints a string and exits

\*/

std\* FromFileToList(std\* head, FILE\* in);

/\*

Function name:FromFileToList

Input:struct and file

Output: sturct

Function Algorithm: creates a linked list by reading from a file.

\*/

std\* Delete\_Stud(std\* toDel, std\* head);

/\*

Function name:Delete\_Stud

Input:sturct,sturct

Output:sturct

Function Algorithm:delete a specific node from a linked list

\*/

std\* DeleteList(std\* head);

/\*

Function name:DeleteList

Input:sturct

Output:sturct

Function Algorithm: Delete a linked List

\*/

void PrintList(std\* head);

/\*

Function name:

Input:

Output:

Function Algorithm:

\*/

std\* FindMax(std\* head);

/\*

Function name:FindMax

Input:sturct

Output:sturct

Function Algorithm:finds the maximum avg number from a node in a linked list.

\*/

int main()

{

int i;

FILE\* f;

std\* Head = NULL, \* temp;

if ((f = fopen("List1.txt", "rt")) == NULL)

Error\_Msg("input error");

Head = FromFileToList(Head, f);

printf("\nThe list is:");

PrintList(Head);

temp = FindMax(Head);

printf("\n\nthe student with max average:\n");

printf("%s %s %s ",temp->code,temp->name,temp->Dep);

for(i=0;i<3;i++)

printf(" %d ",temp->marks[i]);

printf("\n\nThe list after change:");

Head=Delete\_Stud(FindMax(Head),Head);

PrintList(Head);

Head = DeleteList(Head);

return 0;

}

void Error\_Msg(char\* str)

{

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

exit(1);

}

std\* FromFileToList(std\* head, FILE\* in) {

std\* temp;

char tempname[256];

temp = (std\*)malloc(sizeof(std)); //making a new Node

if (temp == NULL) {

Error\_Msg("no memo"); // if memo allocation failed

}

while (fscanf(in, "%s %s %s %d %d %d", temp->code, tempname, temp->Dep, &temp->marks[0], &temp->marks[1], &temp->marks[2]) != EOF) {

temp->name = (char\*)malloc((strlen(tempname) + 1)\*sizeof(char));

if (temp->name == NULL) Error\_Msg("No Memo");

strcpy(temp->name, tempname);

temp->avg = (temp->marks[0] + temp->marks[1] + temp->marks[2]) / 3.0;

temp->next = head; //the last node in the list, next=NULL (and every other iteration initialized to the next node)

head = temp;

temp = (std\*)malloc(sizeof(std)); //making a new Node

if (temp == NULL) {

printf("no memo"); // if memo allocation failed

DeleteList(head);

getchar();

exit(1);

}

}

return head;

}

void PrintList(std\* head) {

std\* temp;

temp = head;

while (temp != NULL) {

printf("\n%s %s %s %d %d %d", temp->code, temp->name, temp->Dep, temp->marks[0], temp->marks[1], temp->marks[2]);

temp = temp->next;

}

}

std\* FindMax(std\* head) {

float max = 0;

std\* temp, \*maxpointer = NULL; //maxpointer to save the node with the max avg.

temp = head;

while (temp)

{

if (temp->avg > max)

{

max = temp->avg;

maxpointer = temp;

}

temp = temp->next;

}

return maxpointer;

}

std\* Delete\_Stud(std\* toDel,std\* head)

{

if(head == NULL)

return NULL;

if(toDel==head)

{

head = head->next;

free(toDel);

}

else

{

std\* temp = head;

while (temp->next != toDel) //we seek the node before toDel

{

temp = temp->next;

}

//temp->next==toDel

temp->next=toDel->next;

free(toDel);

}

return head;

}

std\* DeleteList(std\* head) {

std\* temp;

while (head) { //delete the nodes

temp = head;

head = head->next;

free(temp);

}

return head;

}