**דוח מעבדה 5**

**שמות מגישים:**

סער ויקטור – 312392822

אילון בן סימון – 312162951

**תרגיל 1**

קוד התכנית:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include<conio.h>

#define N 5

//struction decleration

typedef struct Item

{

int Code;

char Name[11];

struct Item \*next;

}Item;

int main()

{

int i;

Item \*Head = NULL, \*temp;

for (i = 1; i <= N; i++) //creating nodes and scanning data to them

{

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

if (temp == NULL)

{

printf("Error! Memory Not Allocated");

exit(1);

}

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

scanf("%d%s", &(temp->Code), temp->Name);

printf("\n");

temp->next = Head;

Head = temp;

} //end loop

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

while (temp !=NULL)

{

printf("%d %s --> ", temp->Code, temp->Name);

temp = temp->next;

}

while (Head!=NULL) //loop to free all the nodes

{

temp = Head;

Head = Head->next;

free(temp);

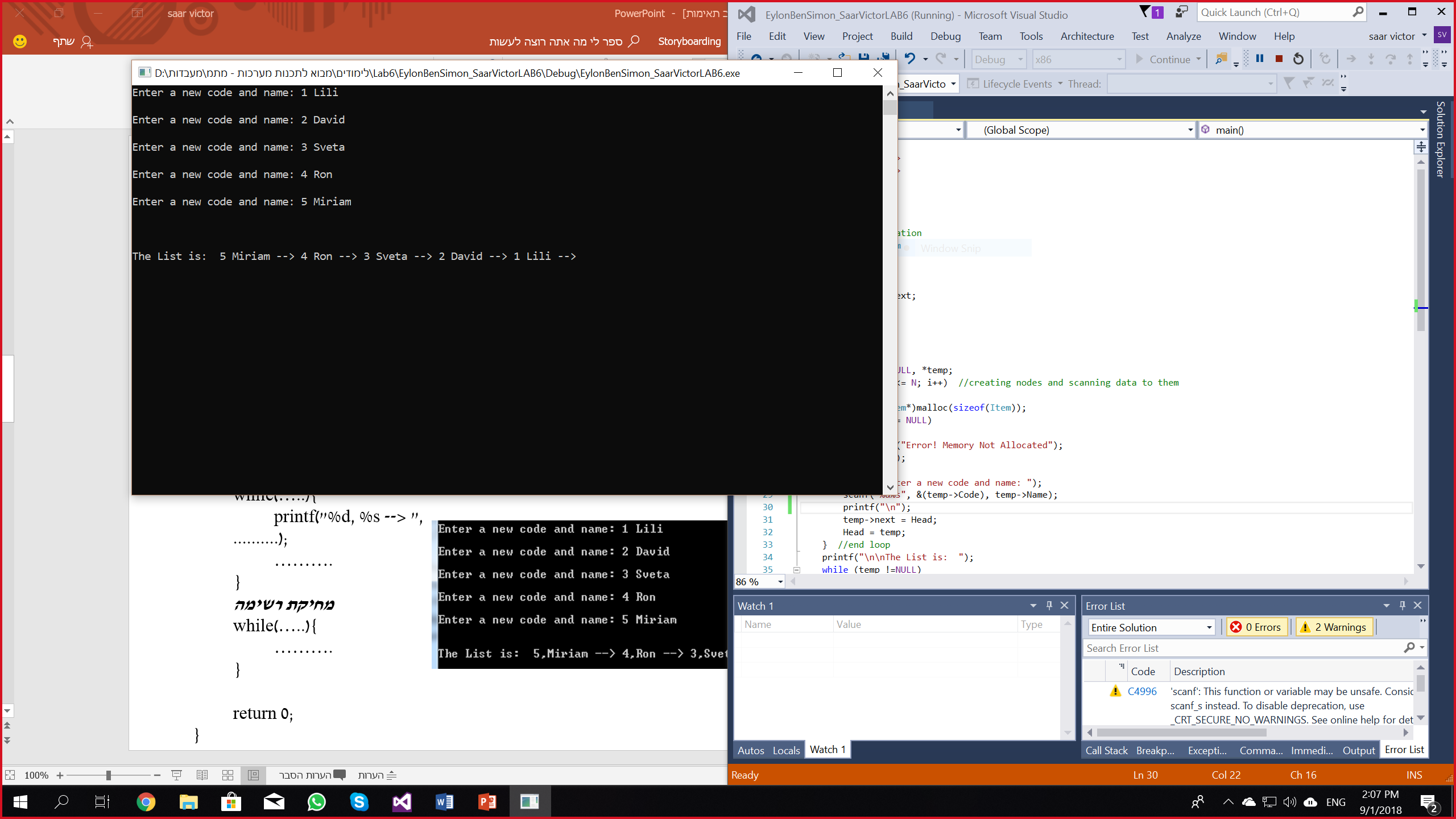
} //end loop

getch();

return 0;

}

פלט:



**תרגיל 2**

קוד התכנית:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <conio.h>

//decleration of struction

typedef struct node

{

char code[11];

char \*name;

char Dep[4];

int marks[3];

float avg;

struct node\* next;

}std;

//decleration of functions

void Error\_Msg(char\* str);

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

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

void DeleteList(std\* head);

void PrintList(std\* head);

std\* FindMax(std\* head);

int main()

{

int i;

FILE \*f;

std \*Head = NULL, \*temp;

if ((f = fopen("List1.txt", "rt")) == NULL) //openning and checking the file to read

Error\_Msg("input error");

Head = FromFileToList(Head, f);

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

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:\n");

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

PrintList(Head);

DeleteList(Head);

Head = NULL;

getch();

return 0;

}

void Error\_Msg(char\* str)

{

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

exit(1);

}

//the function scans the file, build nodes and fill them witht the relevant data

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

{

std\* temp;

char tempName[256];

while (!feof(in)) //creating nodes and scanning data to them

{

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

if (temp == NULL)

{

DeleteList(head);

printf("Error! Memory Not Allocated");

exit(1);

}

fscanf(in, "%s %s %s %d %d %d", temp->code, tempName, temp->Dep, &temp->marks[0], &temp->marks[1], &temp->marks[2]);

temp->name = (char\*)malloc((strlen(tempName) + 1)); //memory allocating the name

if (temp->name == NULL)

{

DeleteList(temp->name);

printf("Error! Memory Not Allocated");

exit(1);

}

strcpy(temp->name, tempName); //copying the data from the temp string to the proper string

temp->next = head;

head = temp;

} //end loop

return head;

}

//the function deletes the node it's being sent: "toDel"

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

{

std\*temp = head;

if (head == NULL)

return NULL;

if (toDel == head) //incase the intended node is the first

{

head = head->next;

free(toDel->name);

free(toDel);

}

else

{

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

{

if (temp->next == toDel) //checks if the next node is the one to delete

{

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

free(toDel->name);

free(toDel);

break;

}

temp = temp->next;

}

}

return head;

}

//the function deletes all node in the linked list

void DeleteList(std\* head)

{

std\*temp;

while (head != NULL)

{

temp = head;

head = head->next;

free(temp->name);

free(temp);

}

}

//the function prints the entire linked list

void PrintList(std\* head)

{

std\* temp;

temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

}

//the function calculates all the average marks and finds the biggest one

std\* FindMax(std\* head)

{

std\* temp = head;

std\* tempMaxGrade = head;

while (temp != NULL) //loop to calculate all the average marks and to find the maximum grade of the linked list

{

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

if (temp->avg > tempMaxGrade->avg)

tempMaxGrade = temp;

temp = temp->next;

} //end loop

temp = head;

return tempMaxGrade;

}

פלט 1 (List.txt): פלט 2 (List1.txt):

