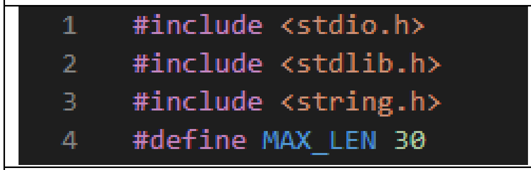
MARMARA UNIVERSITY COMPUTER ENGINEERING
CSE2225 DATA STRUCTURES
REPORT OF PROJECT 2
JUNE 2021

PROFESSOR	MURAT CAN GANİZ

GROUP MEMBERS

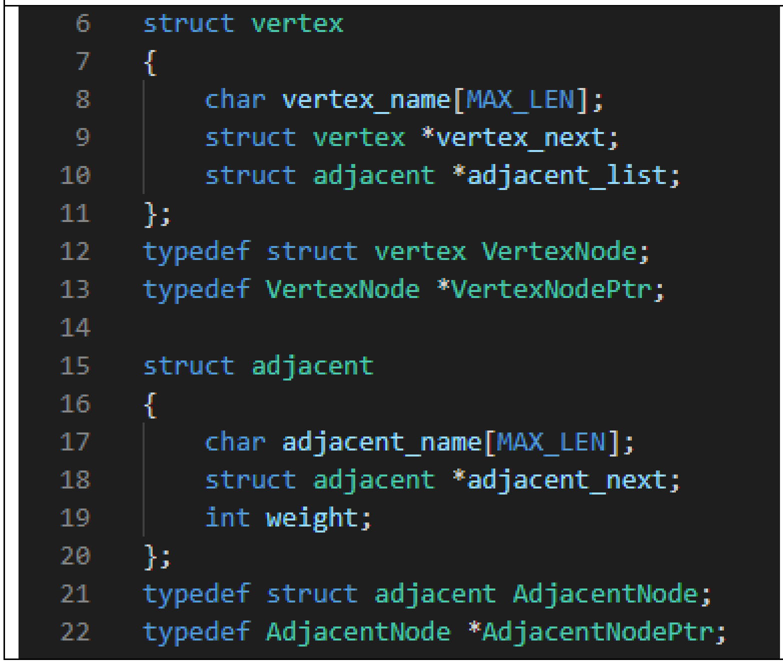
ELİF NUR	KEMİKSİZ	100217006
NESRİN	ŞİMŞEK	150119664
REYTA GÜL	MURAN	150117028

INTRODUCTION



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_LEN 30
```

STRUCTURES



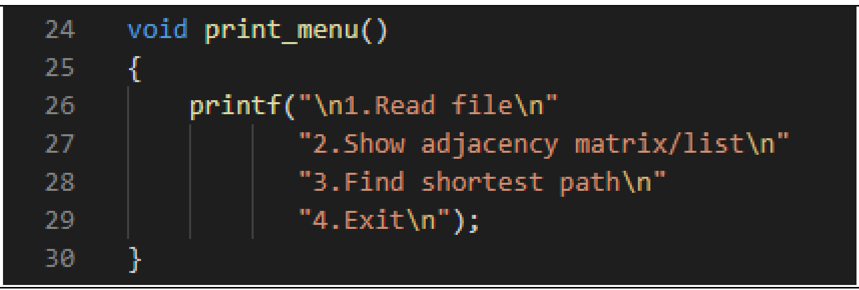
```c
struct vertex
{
    char vertex_name[MAX_LEN];
    struct vertex *vertex_next;
    struct adjacent *adjacent_list;
};
typedef struct vertex VertexNode;
typedef VertexNode *VertexNodePtr;

struct adjacent
{
    char adjacent_name[MAX_LEN];
    struct adjacent *adjacent_next;
    int weight;
};
typedef struct adjacent AdjacentNode;
typedef AdjacentNode *AdjacentNodePtr;
```

- There are 2 structures that hold properties and pointers which points to their next nodes or linked lists derived from other structures.

FUNCTIONS

1. FUNCTION print_menu

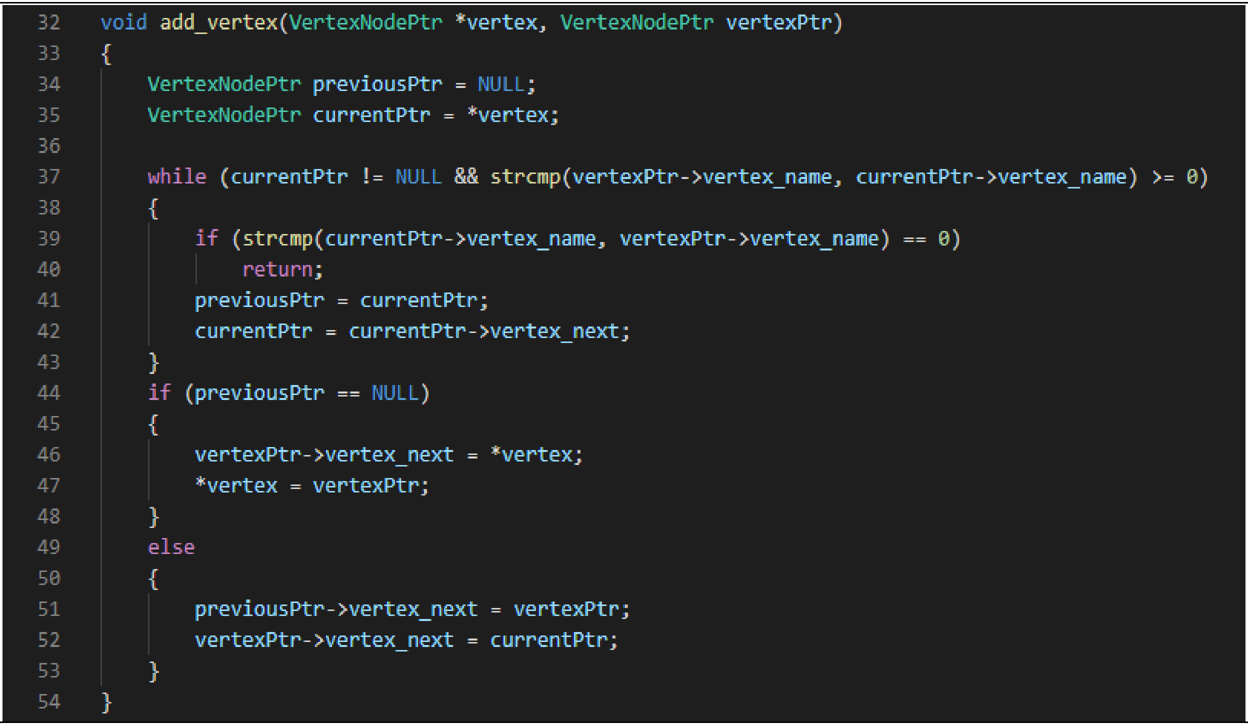


```c
void print_menu()
{
    printf("\n1.Read file\n"
           "2.Show adjacency matrix/list\n"
           "3.Find shortest path\n"
           "4.Exit\n");
}
```

FUNCTION:

- Prints the menu.

2. FUNCTION add_vertex



```c
void add_vertex(VertexNodePtr *vertex, VertexNodePtr vertexPtr)
{
    VertexNodePtr previousPtr = NULL;
    VertexNodePtr currentPtr = *vertex;

    while (currentPtr != NULL && strcmp(vertexPtr->vertex_name, currentPtr->vertex_name) >= 0)
    {
        if (strcmp(currentPtr->vertex_name, vertexPtr->vertex_name) == 0)
            return;
        previousPtr = currentPtr;
        currentPtr = currentPtr->vertex_next;
    }
    if (previousPtr == NULL)
    {
        vertexPtr->vertex_next = *vertex;
        *vertex = vertexPtr;
    }
    else
    {
        previousPtr->vertex_next = vertexPtr;
        vertexPtr->vertex_next = currentPtr;
    }
}
```

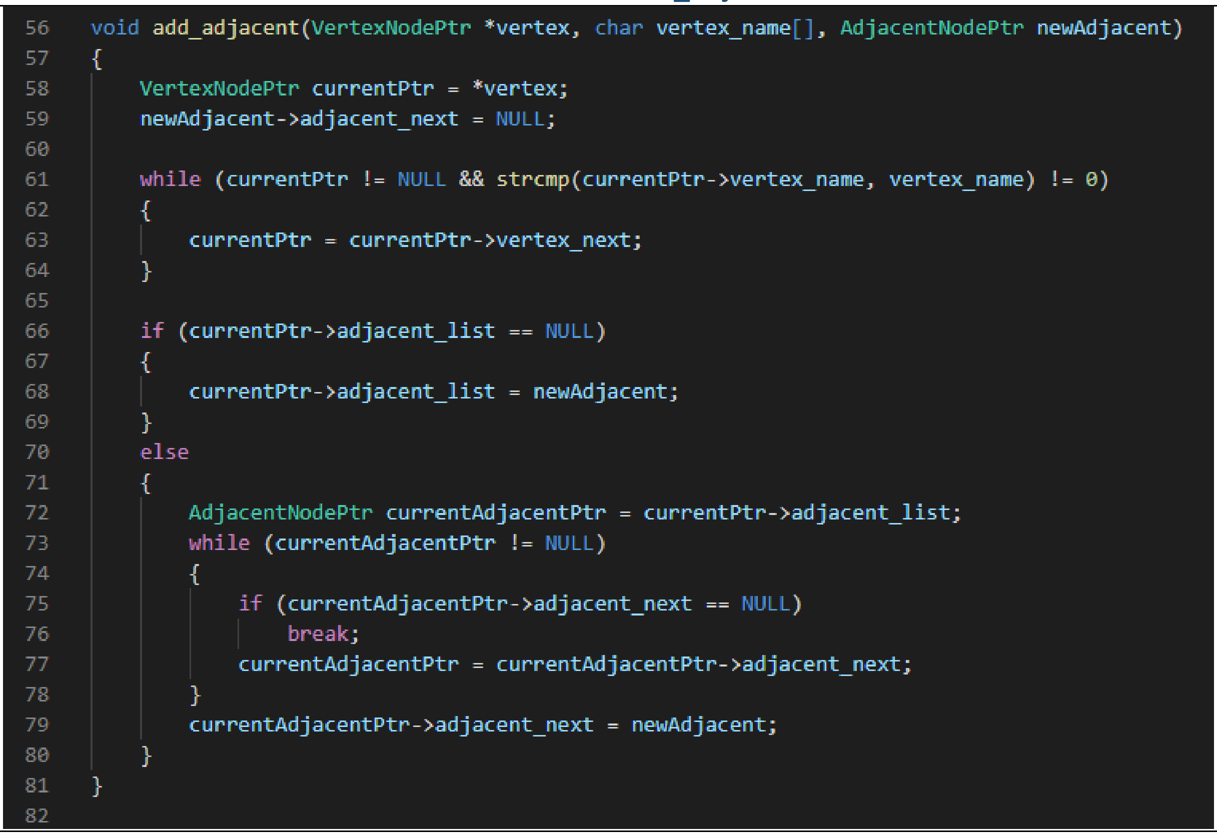
PARAMETERS:

- VertexNodePtr *vertex: Takes the start pointer of vertices as an address.
- VertexNodePtr vertexPtr: The new vertex which will be added to vertex list.

FUNCTION:

- When the user selects 1 from the menu, inserts related vertex from input.txt file to the correct place in the vertex list.
- If there is already such a vertex, returns and does not add that vertex.

3. FUNCTION add_adjacent



```c
void add_adjacent(VertexNodePtr *vertex, char vertex_name[], AdjacentNodePtr newAdjacent)
{
    VertexNodePtr currentPtr = *vertex;
    newAdjacent->adjacent_next = NULL;

    while (currentPtr != NULL && strcmp(currentPtr->vertex_name, vertex_name) != 0)
    {
        currentPtr = currentPtr->vertex_next;
    }

    if (currentPtr->adjacent_list == NULL)
    {
        currentPtr->adjacent_list = newAdjacent;
    }
    else
    {
        AdjacentNodePtr currentAdjacentPtr = currentPtr->adjacent_list;
        while (currentAdjacentPtr != NULL)
        {
            if (currentAdjacentPtr->adjacent_next == NULL)
                break;
            currentAdjacentPtr = currentAdjacentPtr->adjacent_next;
        }
        currentAdjacentPtr->adjacent_next = newAdjacent;
    }
}

```

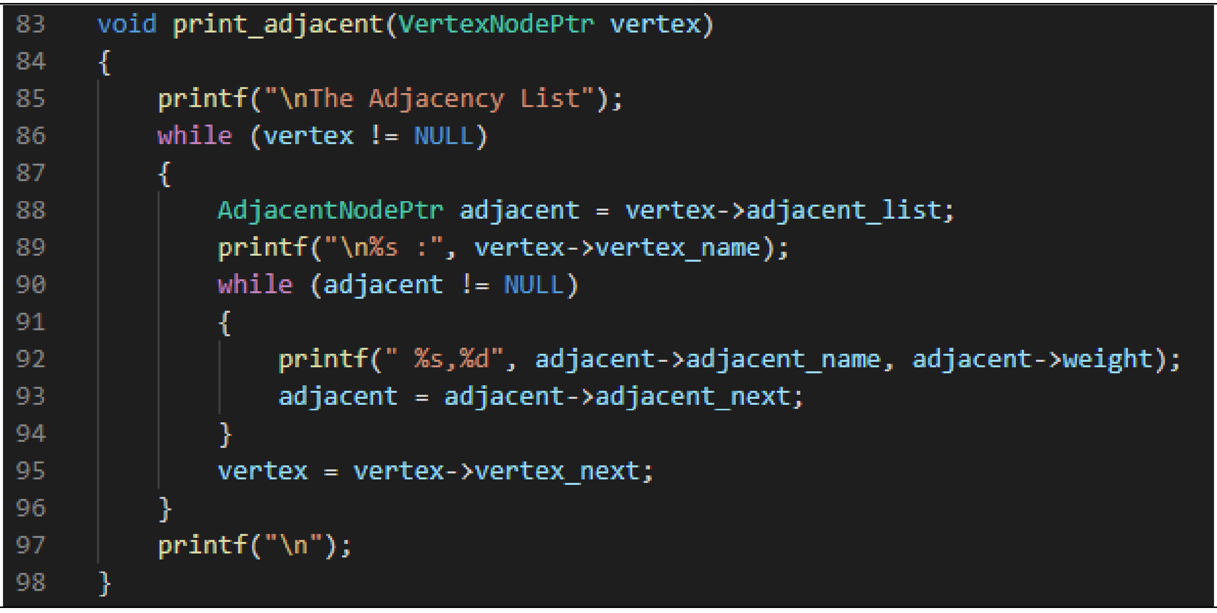
PARAMETERS:

- VertexNodePtr *vertex: Takes the start pointer of vertices as an address.
- char vertex_name[ ]: Takes name of related vertex.
- AdjacentNodePtr newAdjacent: The new adjacent which will be added to adjacent list of related vertex.

FUNCTION:

- When the user selects 1 from the menu, inserts related adjacent from input.txt file to the correct place in the adjacent list of related vertex.

4. FUNCTION print_adjacent



```c
void print_adjacent(VertexNodePtr vertex)
{
    printf("\nThe Adjacency List");
    while (vertex != NULL)
    {
        AdjacentNodePtr adjacent = vertex->adjacent_list;
        printf("\n%s :", vertex->vertex_name);
        while (adjacent != NULL)
        {
            printf(" %s,%d", adjacent->adjacent_name, adjacent->weight);
            adjacent = adjacent->adjacent_next;
        }
        vertex = vertex->vertex_next;
    }
    printf("\n");
}
```

PARAMETERS:

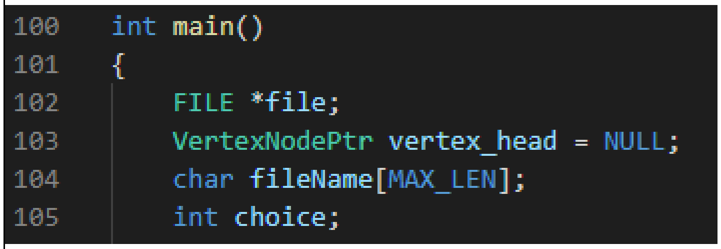
- VertexNodePtr vertex: Takes the start pointer of vertices.

FUNCTION:

- When the user selects 2 from the menu, prints the adjacency list.

MAIN FUNCTION

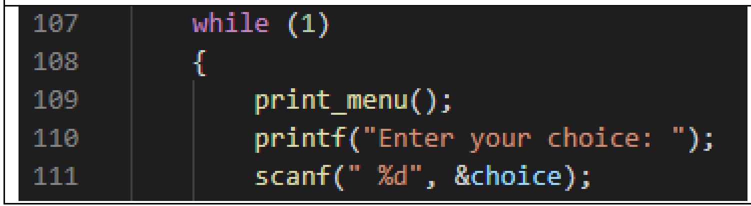
GENERAL PARAMETERS



```c
int main()
{
    FILE *file;
    VertexNodePtr vertex_head = NULL;
    char fileName[MAX_LEN];
    int choice;
```

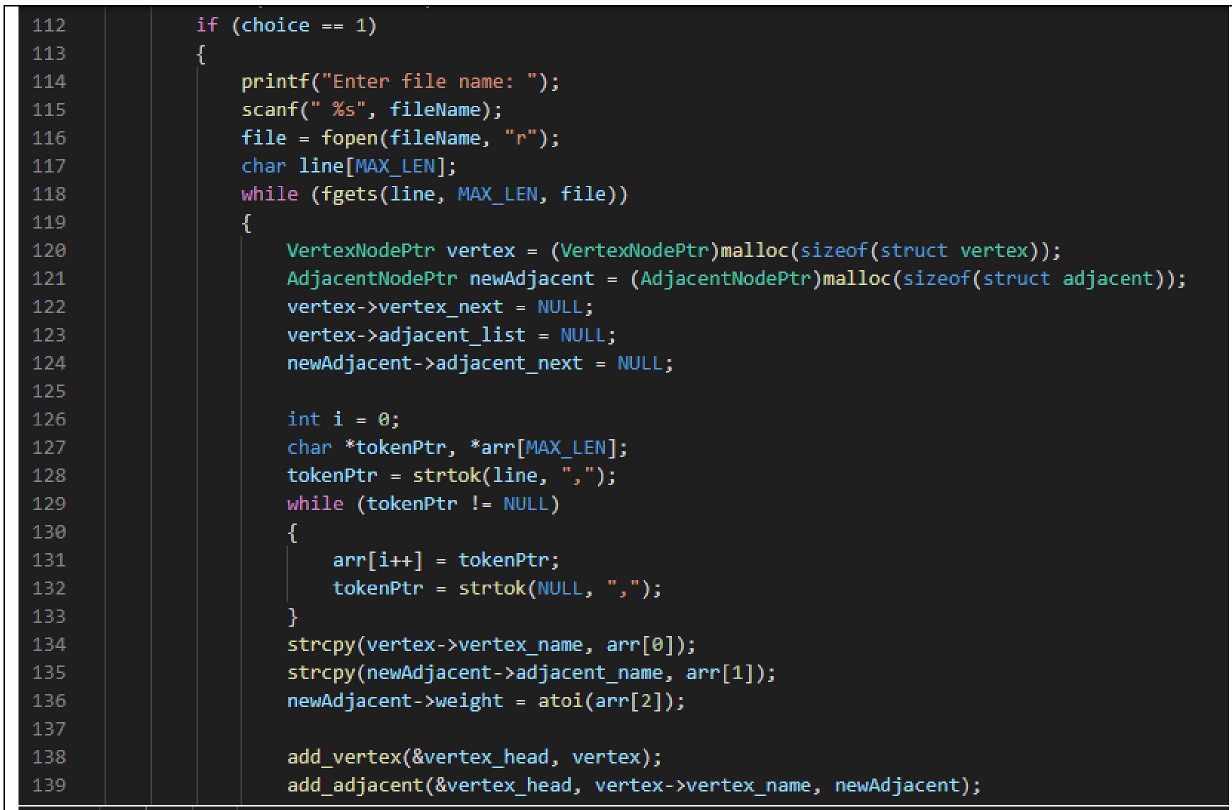
- FILE *file: The file pointer.
- VertexNodePtr vertex_head: Start pointer of vertex linked list.
- char fileName[MAX_LEN]: A string for the file name.
- int choice: An integer for choices in the menu.

USER INPUT-OUTPUT



```c
    while (1)
    {
        print_menu();
        printf("Enter your choice: ");
        scanf(" %d", &choice);
```

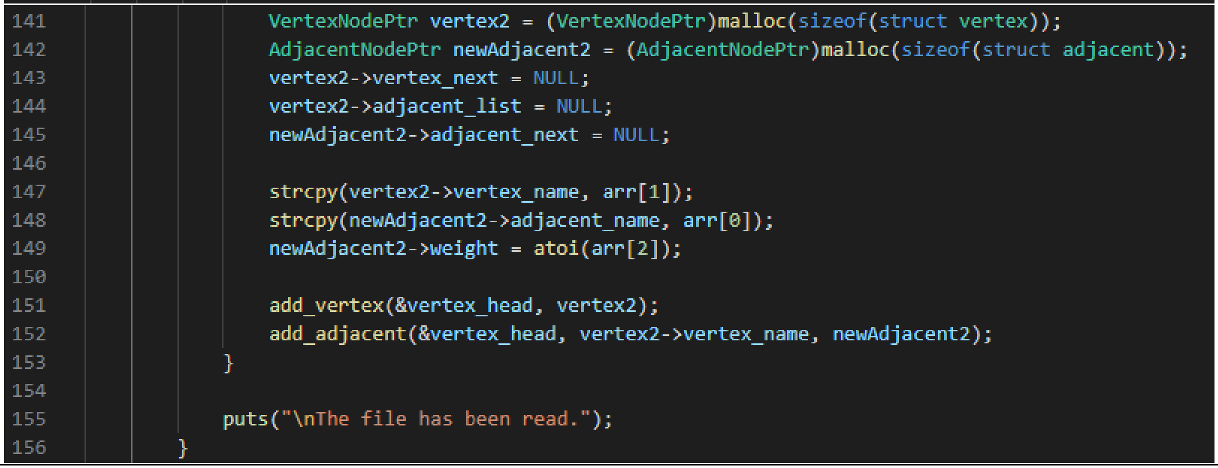
At first, calls print_menu to print the menu.



```c
if (choice == 1)
{
    printf("Enter file name: ");
    scanf(" %s", fileName);
    file = fopen(fileName, "r");
    char line[MAX_LEN];
    while (fgets(line, MAX_LEN, file))
    {
        VertexNodePtr vertex = (VertexNodePtr)malloc(sizeof(struct vertex));
        AdjacentNodePtr newAdjacent = (AdjacentNodePtr)malloc(sizeof(struct adjacent));
        vertex->vertex_next = NULL;
        vertex->adjacent_list = NULL;
        newAdjacent->adjacent_next = NULL;

        int i = 0;
        char *tokenPtr, *arr[MAX_LEN];
        tokenPtr = strtok(line, ",");
        while (tokenPtr != NULL)
        {
            arr[i++] = tokenPtr;
            tokenPtr = strtok(NULL, ",");
        }
        strcpy(vertex->vertex_name, arr[0]);
        strcpy(newAdjacent->adjacent_name, arr[1]);
        newAdjacent->weight = atoi(arr[2]);

        add_vertex(&vertex_head, vertex);
        add_adjacent(&vertex_head, vertex->vertex_name, newAdjacent);
```



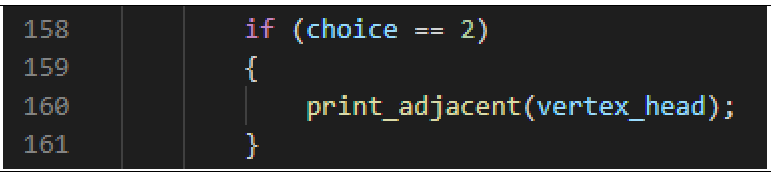
```c
        VertexNodePtr vertex2 = (VertexNodePtr)malloc(sizeof(struct vertex));
        AdjacentNodePtr newAdjacent2 = (AdjacentNodePtr)malloc(sizeof(struct adjacent));
        vertex2->vertex_next = NULL;
        vertex2->adjacent_list = NULL;
        newAdjacent2->adjacent_next = NULL;

        strcpy(vertex2->vertex_name, arr[1]);
        strcpy(newAdjacent2->adjacent_name, arr[0]);
        newAdjacent2->weight = atoi(arr[2]);

        add_vertex(&vertex_head, vertex2);
        add_adjacent(&vertex_head, vertex2->vertex_name, newAdjacent2);
    }

    puts("\nThe file has been read.");
}
```

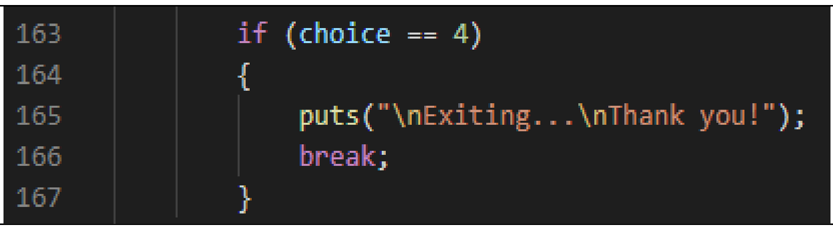
Asks user the file name in order to be read, for every iteration reads the file line by line, adds vertices and adjacents bidirectionally.



```c
if (choice == 2)
{
    print_adjacent(vertex_head);
}
```

Prints the adjacency list.

Choice 3 is not working!!!

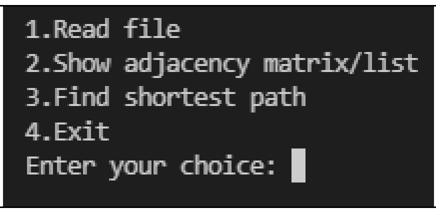


```c
    if (choice == 4)
    {
        puts("\nExiting...\nThank you!");
        break;
    }
```

Exits program.

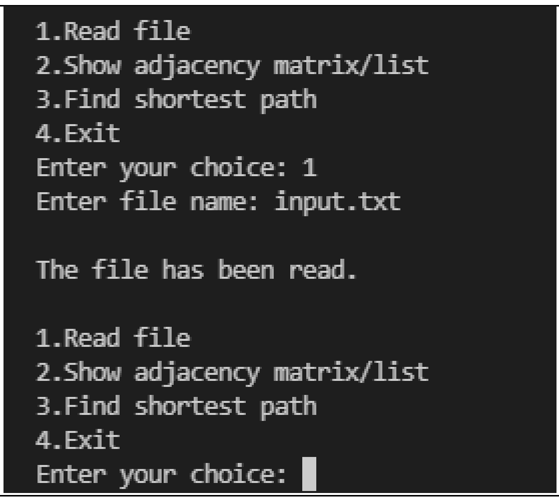
EXECUTIONS

DEFAULT START



```
1.Read file
2.Show adjacency matrix/list
3.Find shortest path
4.Exit
Enter your choice: 
```

CHOICE 1

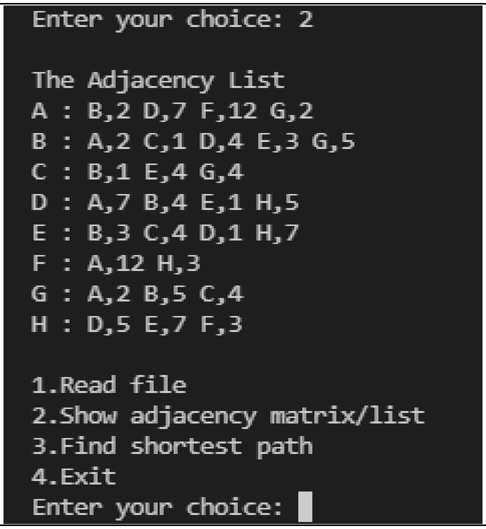


```
1.Read file
2.Show adjacency matrix/list
3.Find shortest path
4.Exit
Enter your choice: 1
Enter file name: input.txt

The file has been read.

1.Read file
2.Show adjacency matrix/list
3.Find shortest path
4.Exit
Enter your choice: 
```

CHOICE 2

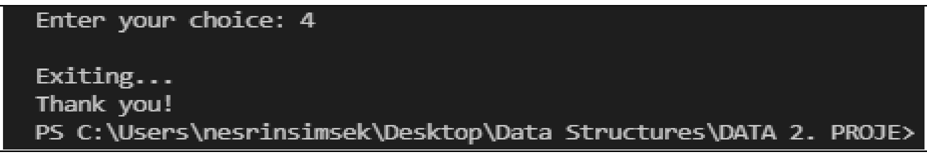


```
Enter your choice: 2

The Adjacency List
A : B,2 D,7 F,12 G,2
B : A,2 C,1 D,4 E,3 G,5
C : B,1 E,4 G,4
D : A,7 B,4 E,1 H,5
E : B,3 C,4 D,1 H,7
F : A,12 H,3
G : A,2 B,5 C,4
H : D,5 E,7 F,3

1.Read file
2.Show adjacency matrix/list
3.Find shortest path
4.Exit
Enter your choice: 
```

CHOICE 4



```
Enter your choice: 4

Exiting...
Thank you!
PS C:\Users\nesrinsimsek\Desktop\Data Structures\DATA 2. PROJE>
```