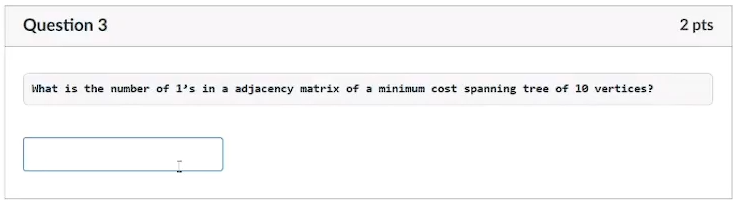
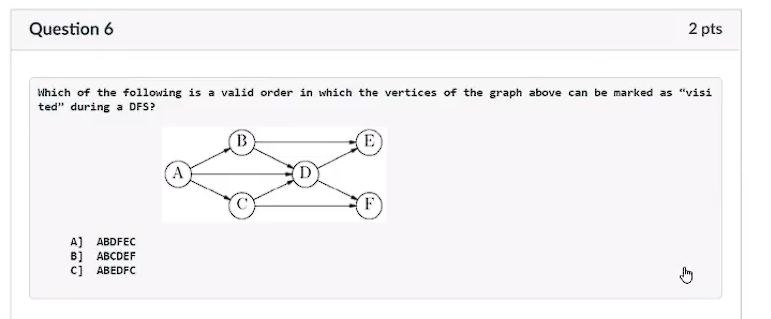
Graphical user interface, text, application, email

Description automatically generated

**D🡪B🡪E🡪A🡪C🡪F**

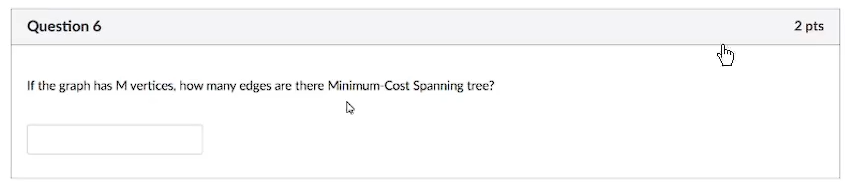


**9 one’s** (not sure pls double check using matrix jd)

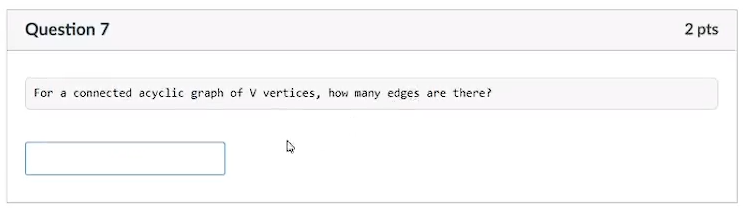


**ABDFEC**, **ABEDFC** (ABCDEF)

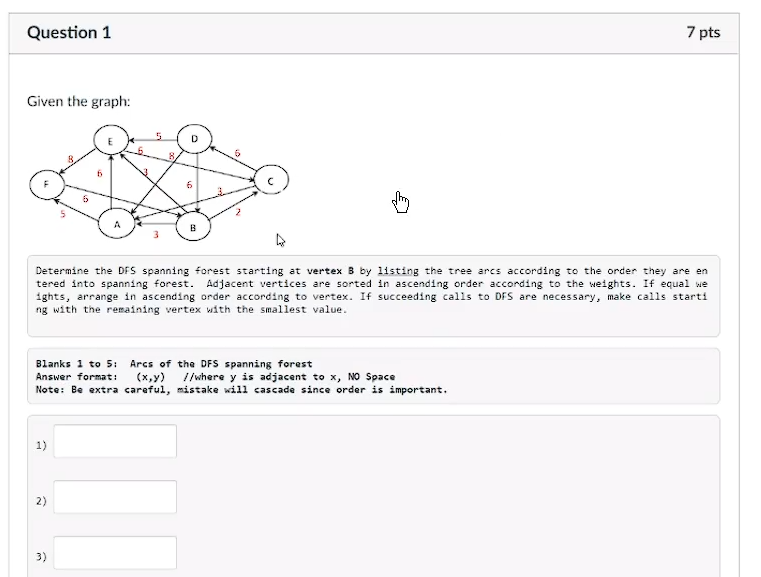
**A and C** (B if invalid order)



**M-1**

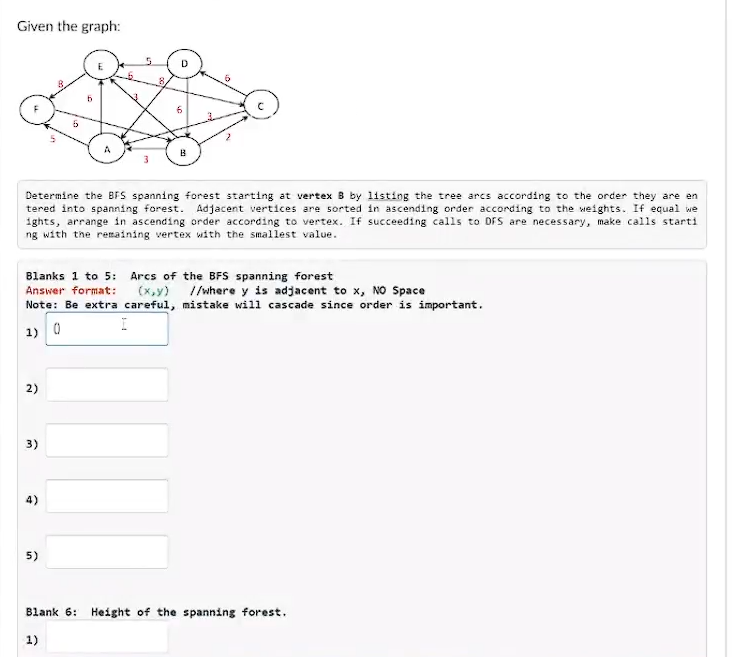


**V-1**



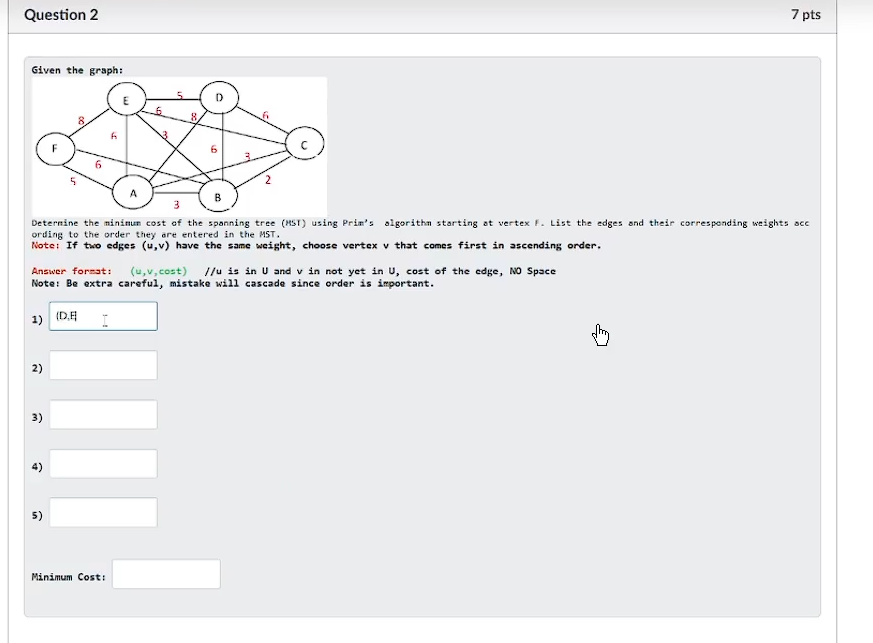
(B, C) (C, A) (A, F) (A, E) (C, D)

**Height**: 3



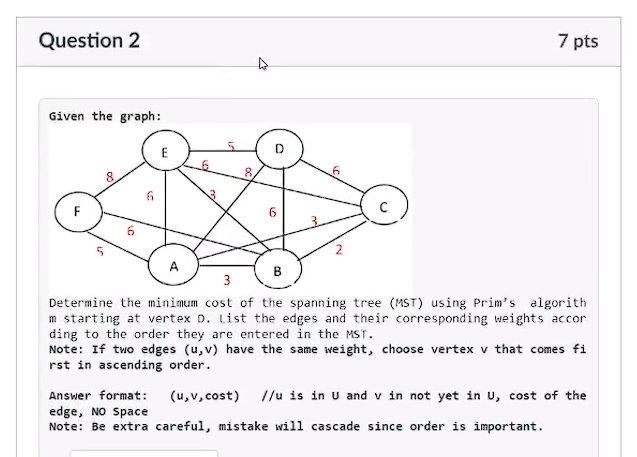
(B, C) (B, A) (B, E) (C, D) (A, F)

**Height:** 2



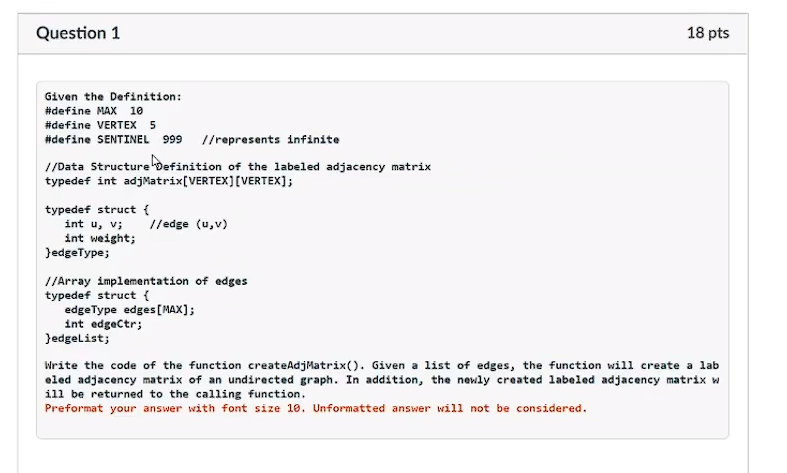
(F, A, 5) (A, B, 3) (B, C, 2) (B, E, 3) (E, D, 5)

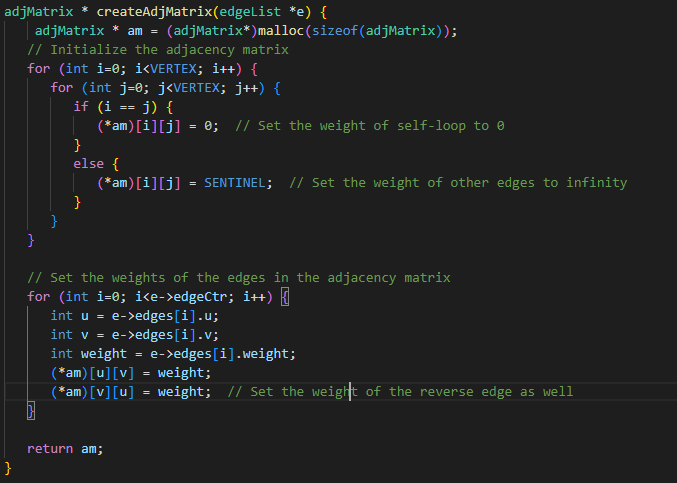
**Minimum cost**: 18



(D, E, 5) (E, B, 3) (B, C, 2) (B, A, 3) (A, F, 5)

**Minimum cost**: 18



given the definition

#define MAX 10

#define VERTEX 5

#define SENTINEL 999

typedef int adjMatrix[VERTEX][VERTEX];

typedef struct {

int u, v;

int weight;

} edgeType;

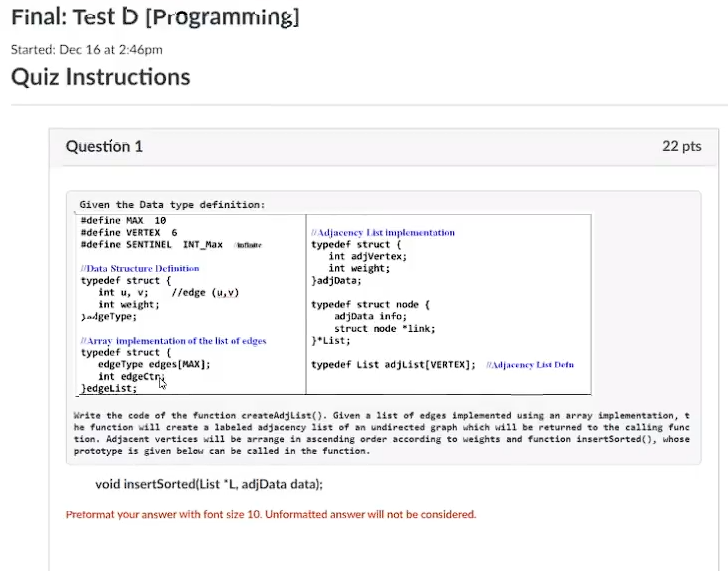
typedef struct {

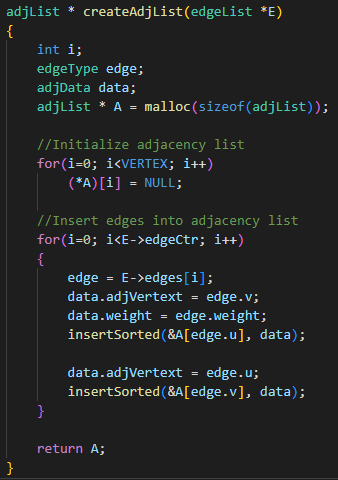
edgeType edges[MAX];

int edgeCtr;

} edgeList;

Write the code of the function createAdjMatrix(). Given a list of edges, the function will create a labeled adjacency matrix of an undirected graph. In addition, the newly created labeled adjacency matrix will be returned to the calling function



given the data type definition

#define MAX 10

#define VERTEX 6

#define SENTINEL INT\_Max

typedef struct {

int u, v;

int weight;

} edgeType;

typedef struct {

edgeType edges[MAX];

int edgeCtr;

} edgeList;

typedef struct {

int adjVertex;

int weight;

} adjData;

typedef struct node {

adjData info;

struct node \*link;

} \*List;

typedef List adjList[VERTEX];

write the code of the function createAdjList(). Given a list of edges implemented using an array implementation, the function will create a labeled list of an undirected graph which will be returned to the calling function. Adjacent vertices will be arranged in ascending order according to weights and function insertSorted(), whose prototype is given as void insertSorted(List \*L, adjData data); can be called in the function.