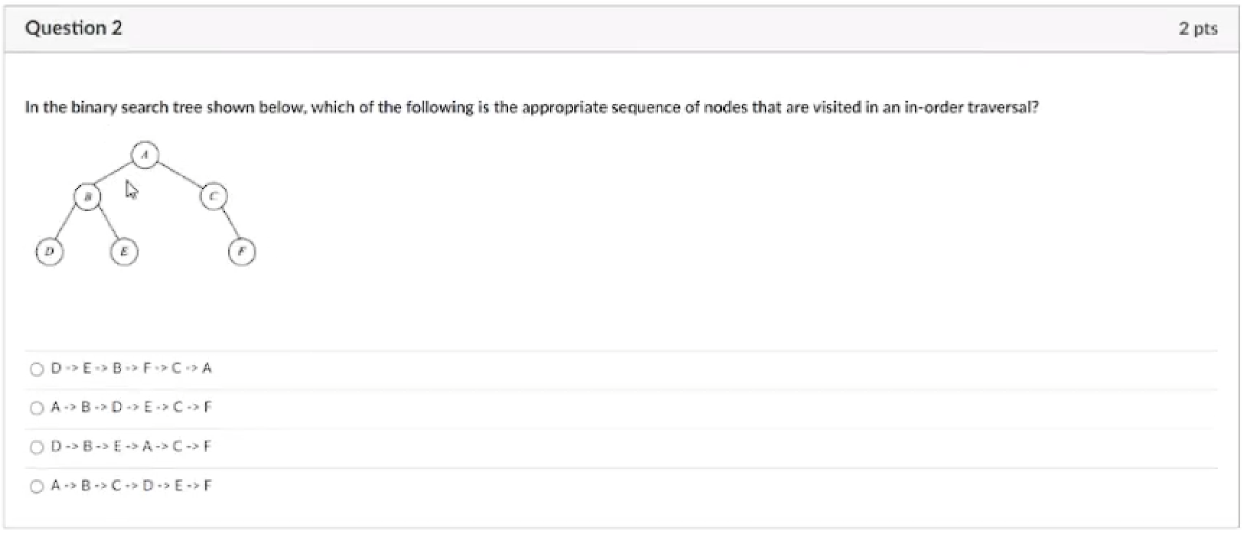
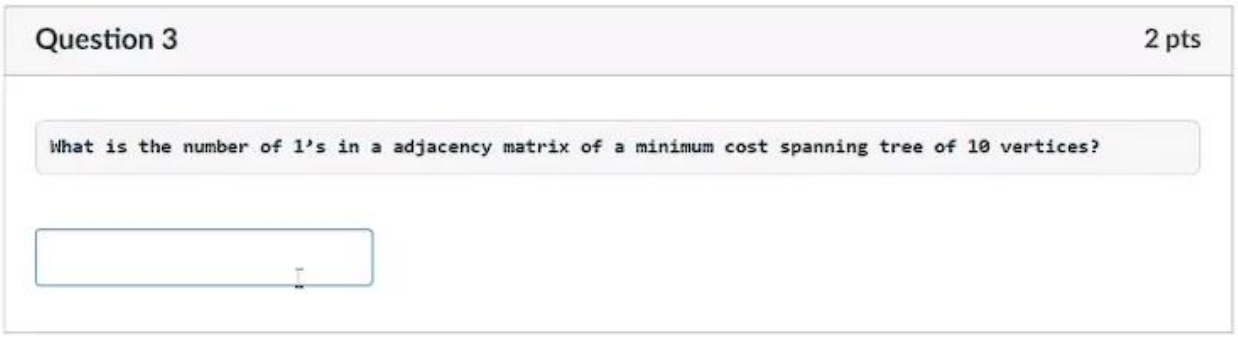

Question 2 2 pts

In the binary search tree shown below, which of the following is the appropriate sequence of nodes that are visited in an in-order traversal?

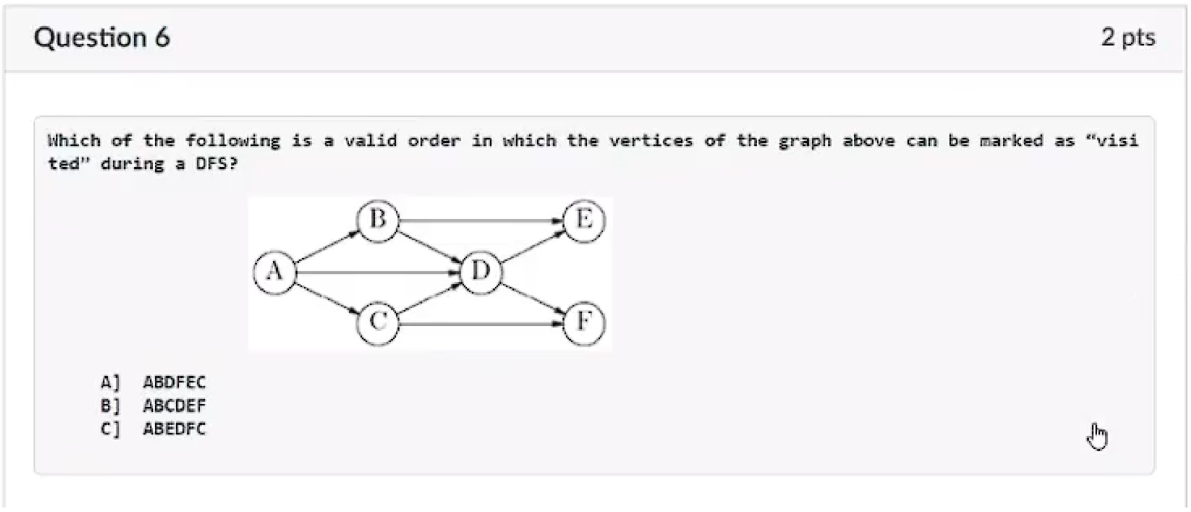
- D -> E -> B -> F -> C -> A
- A -> B -> D -> E -> C -> F
- D -> B -> E -> A -> C -> F
- A -> B -> C -> D -> E -> F

D→B→E→A→C→F


Question 3 2 pts

What is the number of 1's in a adjacency matrix of a minimum cost spanning tree of 10 vertices?

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

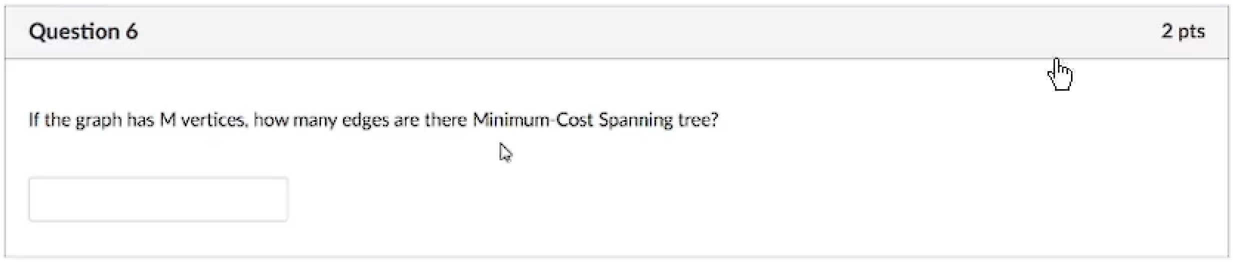

Question 6 2 pts

Which of the following is a valid order in which the vertices of the graph above can be marked as "visited" during a DFS?

A] ABDFEC
B] ABCDEF
C] ABEDFC

ABDFEC, **ABEDFC** (ABCDEF)

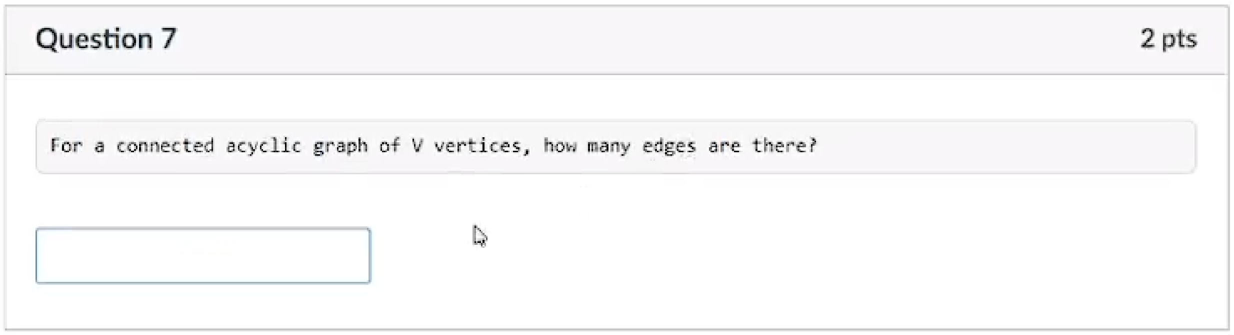
A and C (B if invalid order)


Question 6

2 pts

If the graph has M vertices, how many edges are there Minimum-Cost Spanning tree?

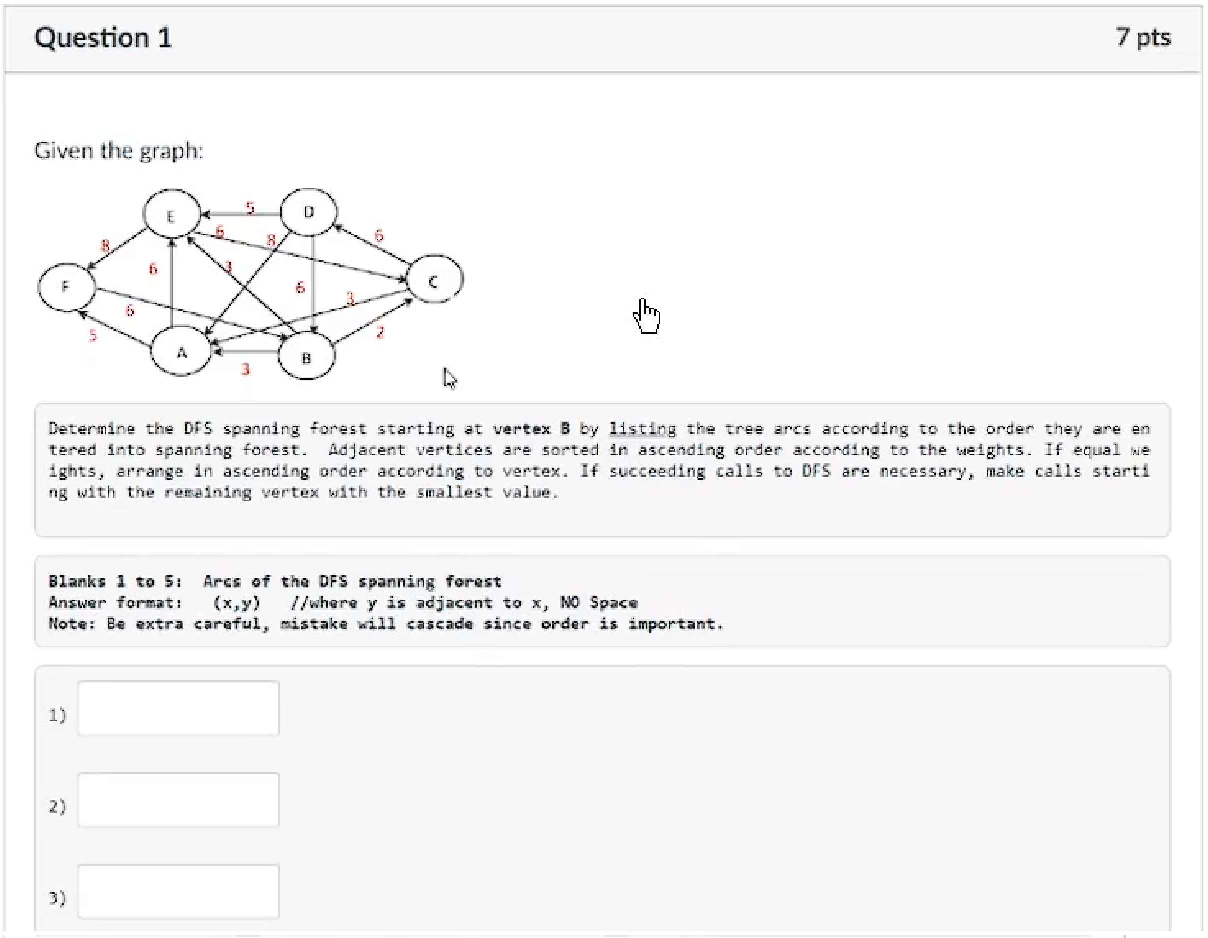
M-1


Question 7

2 pts

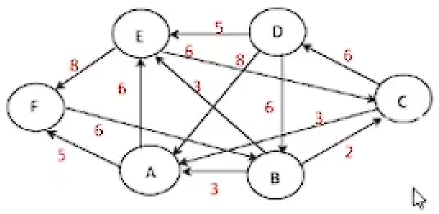
```
For a connected acyclic graph of V vertices, how many edges are there?
```

V-1



Question 1 7 pts

Given the graph:



```
Determine the DFS spanning forest starting at vertex B by listing the tree arcs according to the order they are en
tered into spanning forest.  Adjacent vertices are sorted in ascending order according to the weights. If equal we
ights, arrange in ascending order according to vertex. If succeeding calls to DFS are necessary, make calls starti
ng with the remaining vertex with the smallest value.
```

```
Blanks 1 to 5:  Arcs of the DFS spanning forest
Answer format:   (x,y)   //where y is adjacent to x, NO Space
Note: Be extra careful, mistake will cascade since order is important.
```

1)

2)

3)

1)

2)

3)

4)

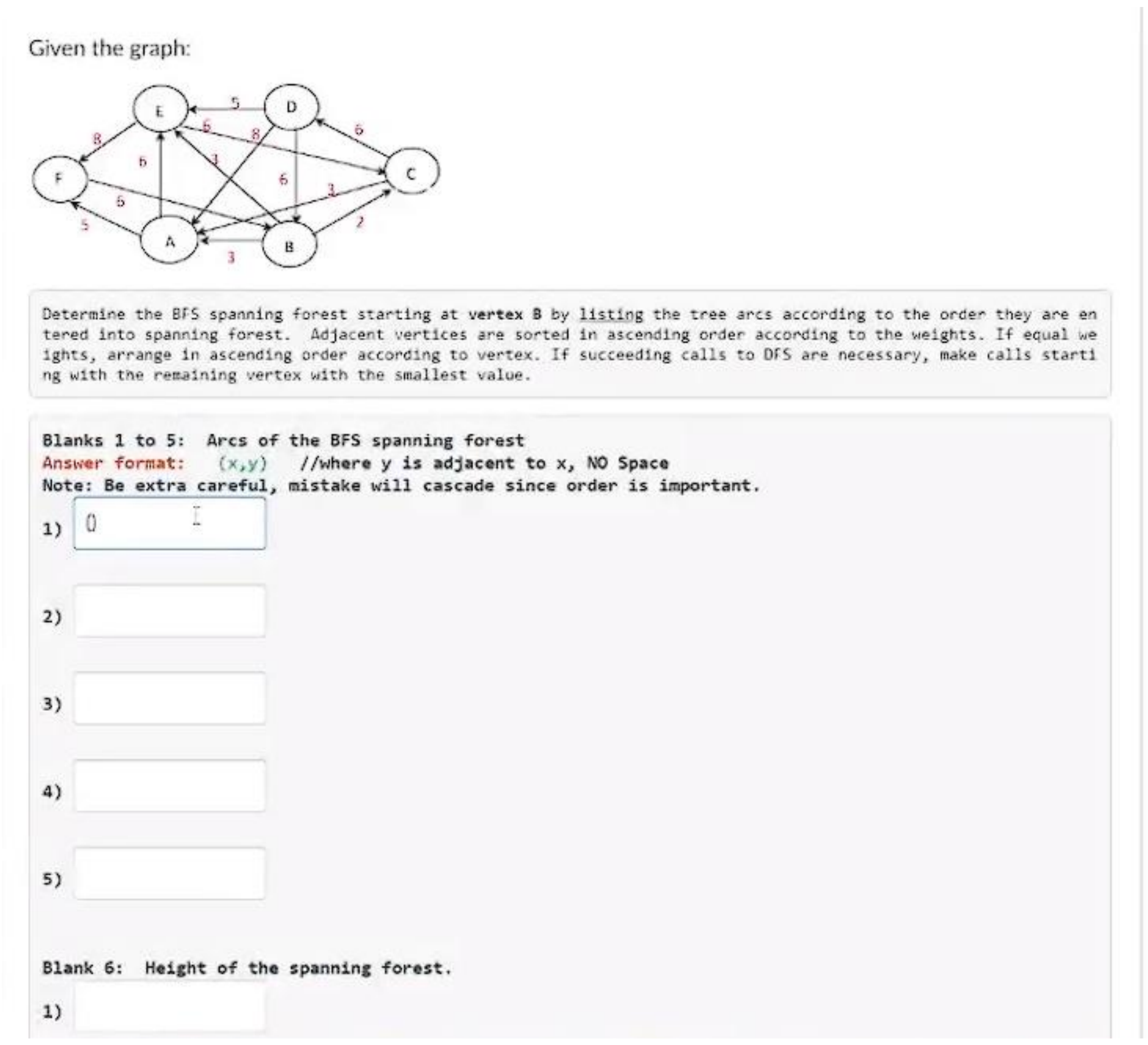
5)

Blank 6: Height of the spanning forest.

1)

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

Height: 3



Given the graph:

Determine the BFS spanning forest starting at **vertex B** by listing the tree arcs according to the order they are entered into spanning forest. Adjacent vertices are sorted in ascending order according to the weights. If equal weights, arrange in ascending order according to vertex. If succeeding calls to DFS are necessary, make calls starting with the remaining vertex with the smallest value.

Blanks 1 to 5: Arcs of the BFS spanning forest
Answer format: (x,y) **//where y is adjacent to x, NO Space**
Note: Be extra careful, mistake will cascade since order is important.

1) ()

2)

3)

4)

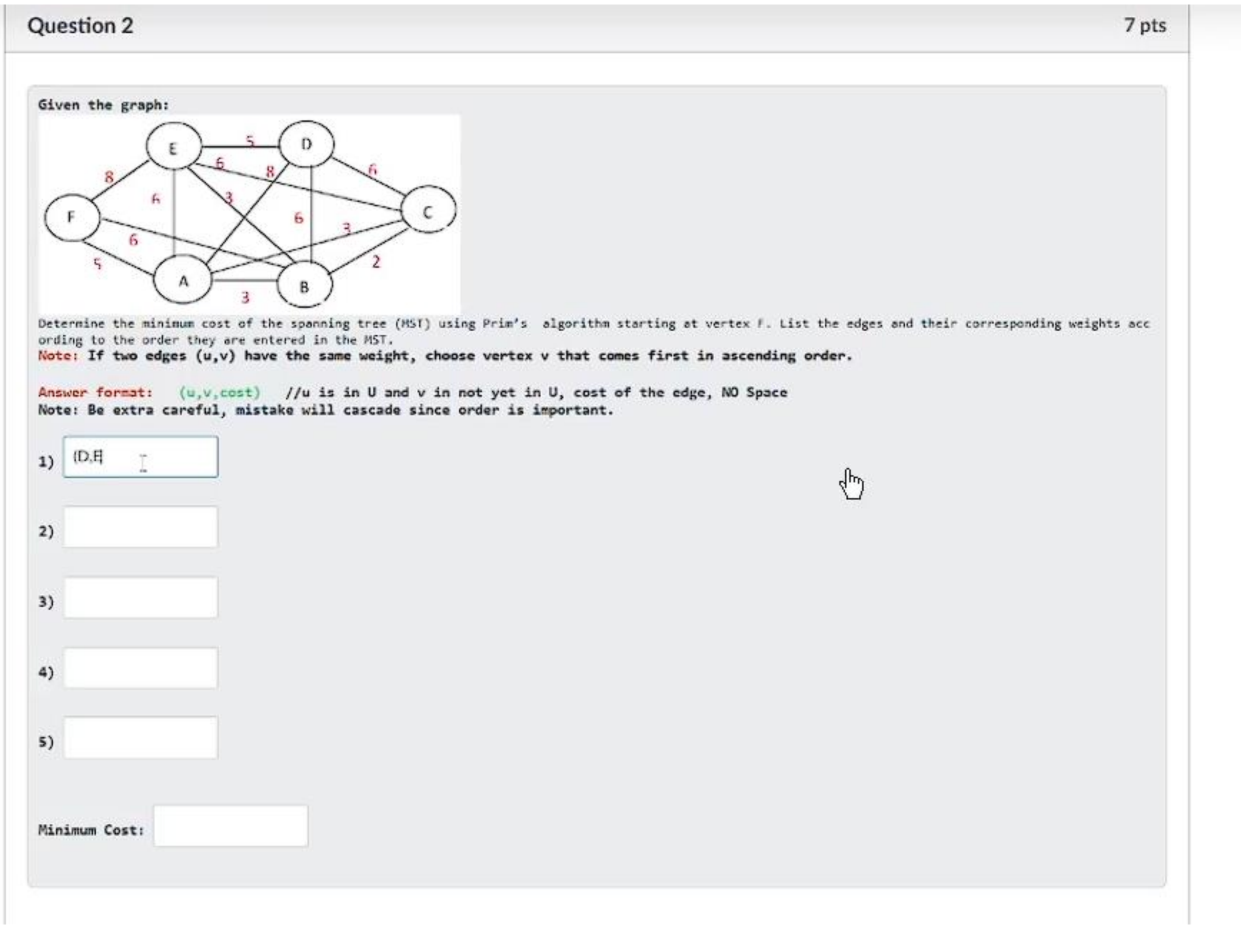
5)

Blank 6: Height of the spanning forest.

1)

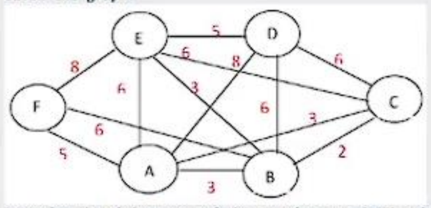
(B, C) (B, A) (B, E) (C, D) (A, F)

Height: 2



Question 2 7 pts

Given the graph:



Determine the minimum cost of the spanning tree (MST) using Prim's algorithm starting at vertex F. List the edges and their corresponding weights according to the order they are entered in the MST.
Note: **If two edges (u,v) have the same weight, choose vertex v that comes first in ascending order.**

Answer format: (u,v,cost) **//u is in U and v in not yet in U, cost of the edge, NO Space**
Note: Be extra careful, mistake will cascade since order is important.

1) (D,E

2)

3)

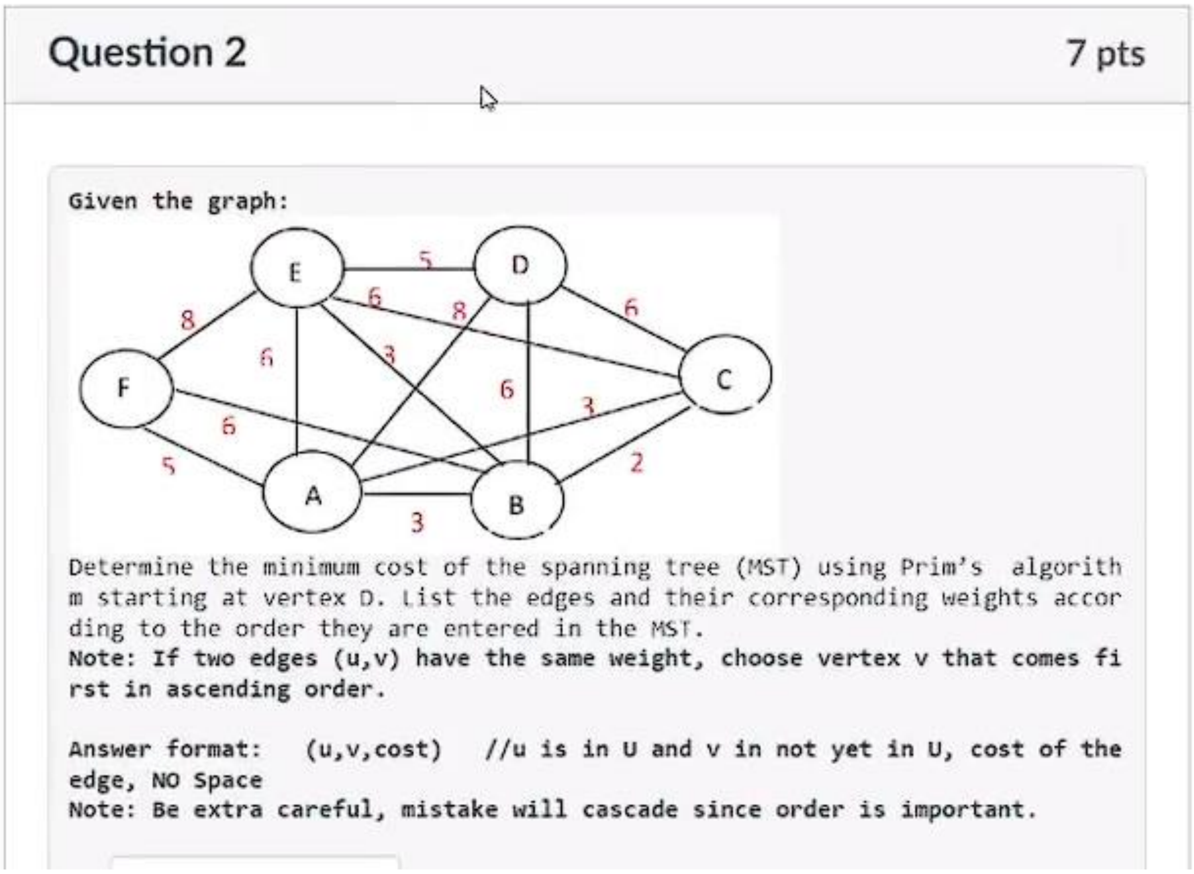
4)

5)

Minimum Cost:

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

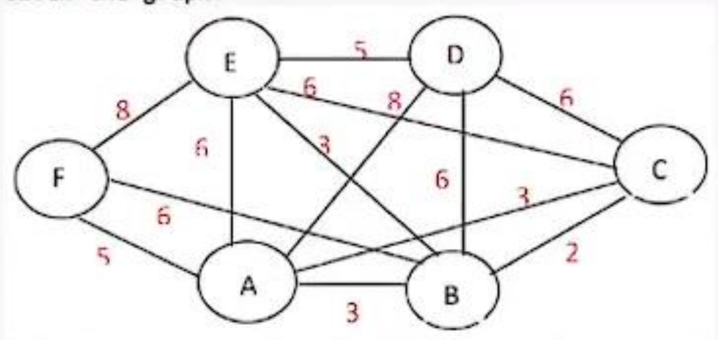
Minimum cost: 18



Question 2

7 pts

Given the graph:

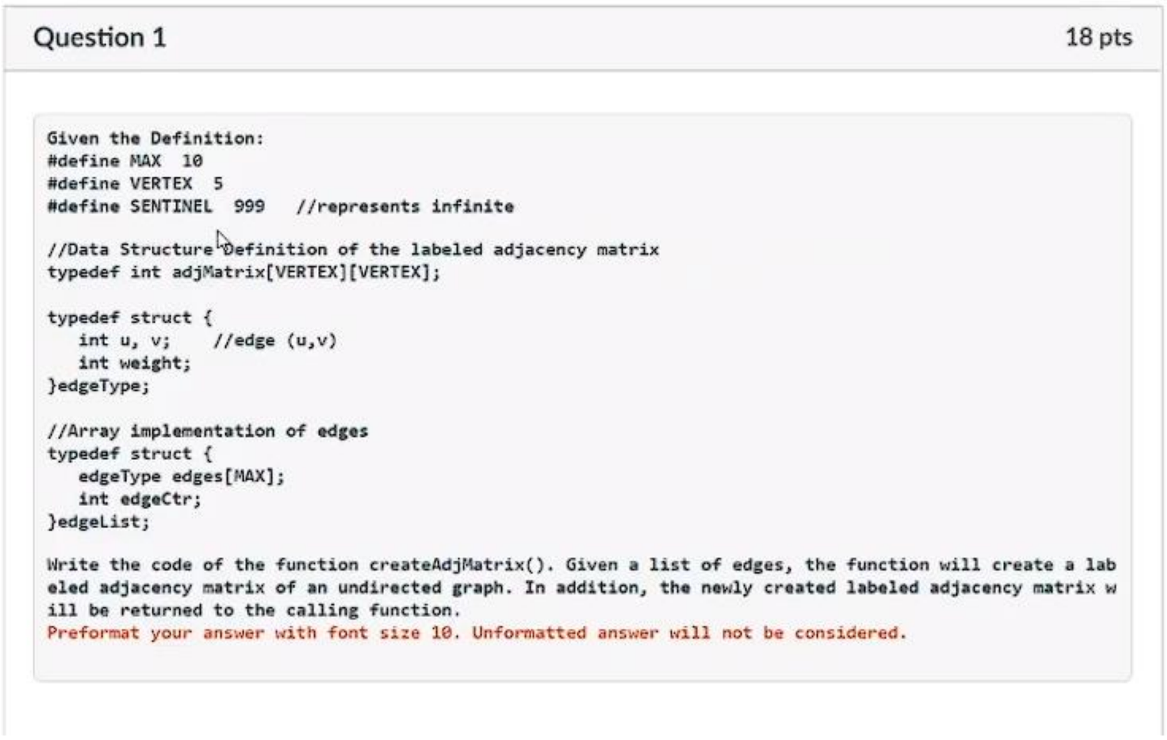


Determine the minimum cost of the spanning tree (MST) using Prim's algorithm starting at vertex D. List the edges and their corresponding weights according to the order they are entered in the MST.
Note: If two edges (u,v) have the same weight, choose vertex v that comes first in ascending order.

Answer format: (u,v,cost) //u is in U and v in not yet in U, cost of the edge, NO Space
Note: Be extra careful, mistake will cascade since order is important.

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

Minimum cost: 18



Question 1 18 pts

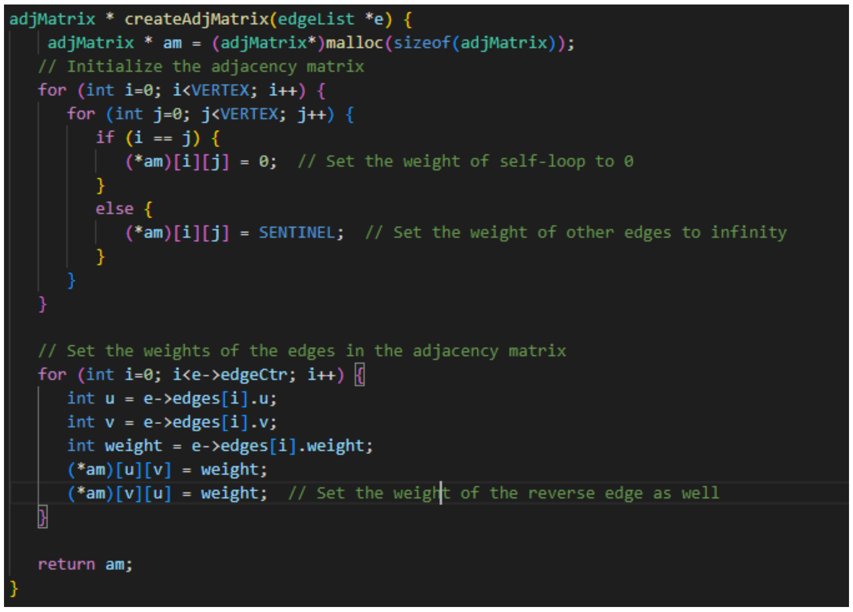
```
Given the Definition:
#define MAX  10
#define VERTEX  5
#define SENTINEL  999   //represents infinite

//Data Structure Definition of the labeled adjacency matrix
typedef int adjMatrix[VERTEX][VERTEX];

typedef struct {
   int u, v;    //edge (u,v)
   int weight;
}edgeType;

//Array implementation of edges
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 lab
eled adjacency matrix of an undirected graph. In addition, the newly created labeled adjacency matrix w
ill be returned to the calling function.
Preformat your answer with font size 10. Unformatted answer will not be considered.
```



```
adjMatrix * createAdjMatrix(edgeList *e) {
    adjMatrix * am = (adjMatrix*)malloc(sizeof(adjMatrix));
   // Initialize the adjacency matrix
   for (int i=0; i<VERTEX; i++) {
      for (int j=0; j<VERTEX; j++) {
         if (i == j) {
            (*am)[i][j] = 0;  // Set the weight of self-loop to 0
         }
         else {
            (*am)[i][j] = SENTINEL;  // Set the weight of other edges to infinity
         }
      }
   }

   // Set the weights of the edges in the adjacency matrix
   for (int i=0; i<e->edgeCtr; i++) {
      int u = e->edges[i].u;
      int v = e->edges[i].v;
      int weight = e->edges[i].weight;
      (*am)[u][v] = weight;
      (*am)[v][u] = weight;  // Set the weight of the reverse edge as well
   }

   return am;
}
```

given the definition

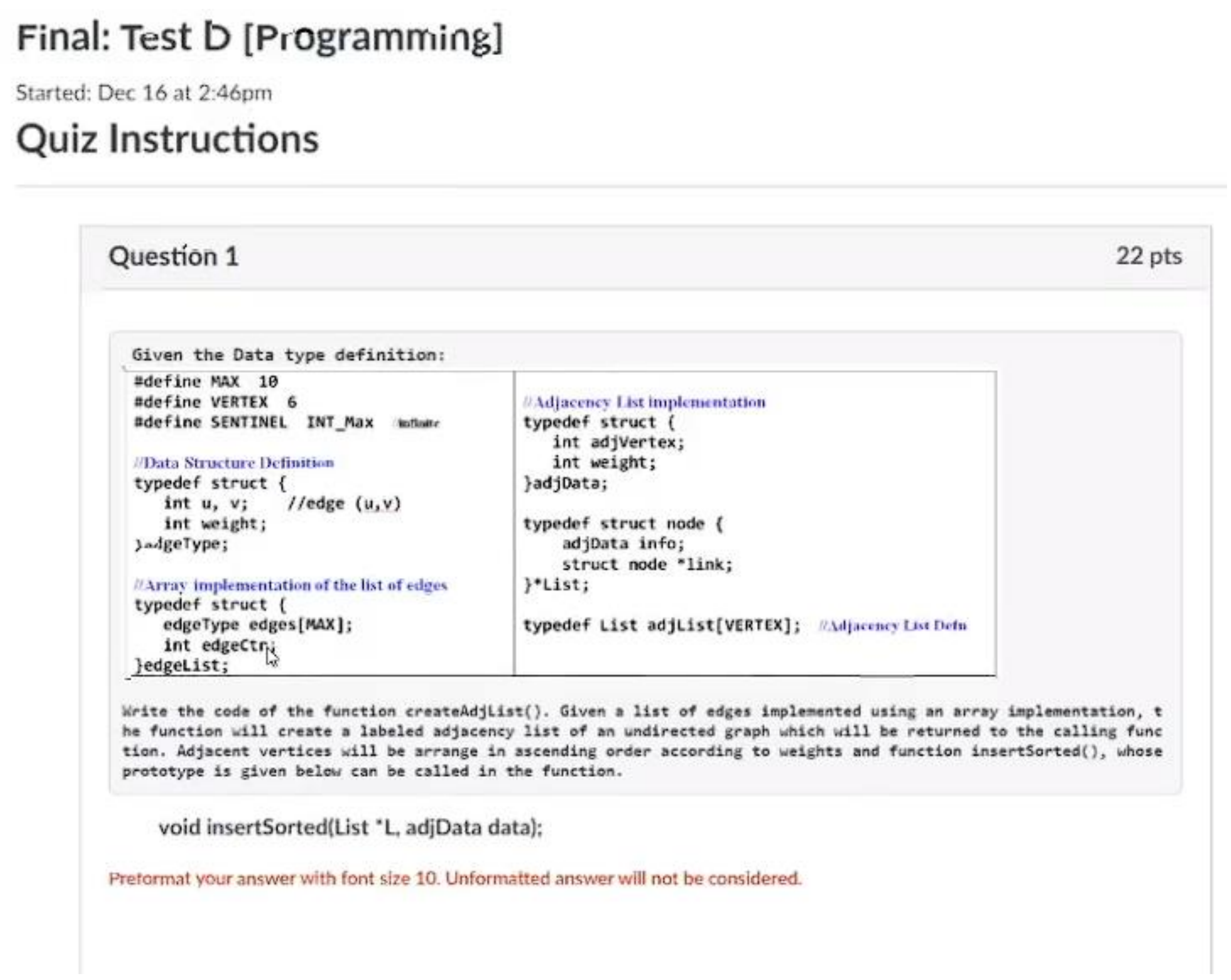
```
#define MAX 10
#define VERTEX 5
#define SENTINEL 999

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



Final: Test D [Programming]

Started: Dec 16 at 2:46pm

Quiz Instructions

Question 1 22 pts

Given the Data type definition:

```
#define MAX  10
#define VERTEX  6
#define SENTINEL  INT_Max   //infinite

//Data Structure Definition
typedef struct {
    int u, v;    //edge (u,v)
    int weight;
}edgeType;

//Array implementation of the list of edges
typedef struct {
    edgeType edges[MAX];
    int edgeCtr;
}edgeList;
```

```
//Adjacency List implementation
typedef struct {
    int adjVertex;
    int weight;
}adjData;

typedef struct node {
     adjData info;
     struct node *link;
}*List;

typedef List adjList[VERTEX];  //Adjacency List Defn
```

Write the code of the function createAdjList(). Given a list of edges implemented using an array implementation, the function will create a labeled adjacency list of an undirected graph which will be returned to the calling function. Adjacent vertices will be arrange in ascending order according to weights and function insertSorted(), whose prototype is given below can be called in the function.

void insertSorted(List *L, adjData data);

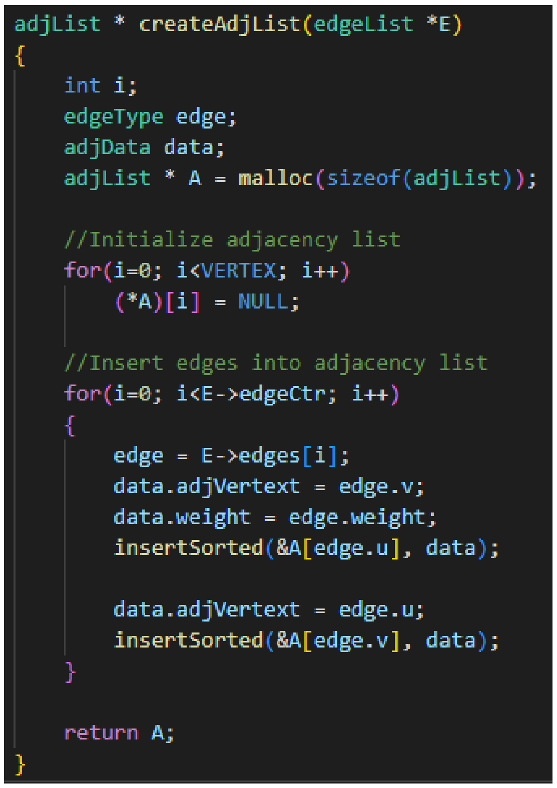
Preformat your answer with font size 10. Unformatted answer will not be considered.

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;



```
adjList * createAdjList(edgeList *E)
{
    int i;
    edgeType edge;
    adjData data;
    adjList * A = malloc(sizeof(adjList));

    //Initialize adjacency list
    for(i=0; i<VERTEX; i++)
        (*A)[i] = NULL;

    //Insert edges into adjacency list
    for(i=0; i<E->edgeCtr; i++)
    {
        edge = E->edges[i];
        data.adjVertext = edge.v;
        data.weight = edge.weight;
        insertSorted(&A[edge.u], data);

        data.adjVertext = edge.u;
        insertSorted(&A[edge.v], data);
    }

    return A;
}
```

```
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.