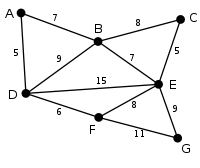
**Exercise 4. Answer Sheet**

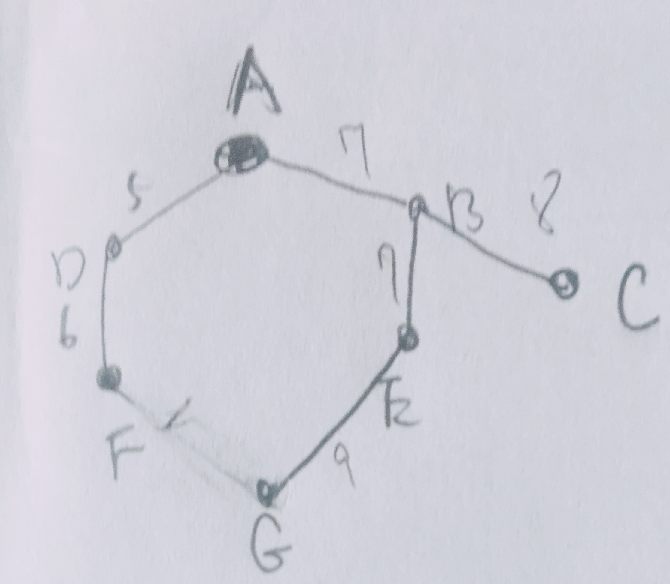
Student's Name: Tomonori Masubuchi Student's ID: s1240078

***Problem 1.*** (*50 points*) Consider the following graph and assume node A as a root.



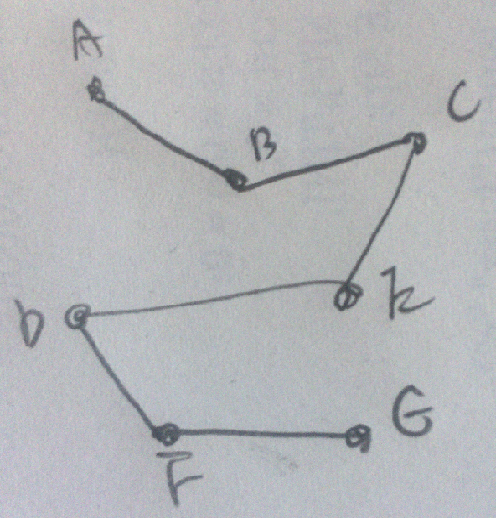
a) Draw a spanning tree obtained by using the Breadth Fist Search (BFS) algorithm.

Put your answer here.

**

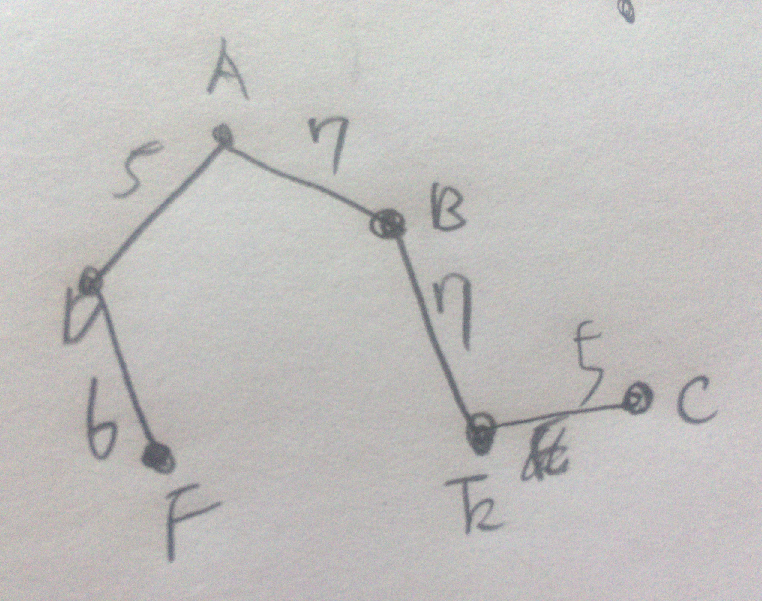
b) Draw a spanning tree obtained by using the Depth First Search (DFS) algorithm.

Put your answer here.

**

c) Draw the minimum spanning tree obtained by the Prim's algorithm.

Put your answer here.

******

***Problem 2.*** *(50 points)*Write a program implementing Kruskal'salgorithm. Upload your source code. Show your input graph and the obtained MST in the space below.

#include<stdio.h>

#define MAX 50

int G[MAX][MAX],n;

typedef struct edge

{

int u,v,w;

}edge;

typedef struct edgelist

{

edge data[MAX];

int n;

}edgelist;

edgelist spanlist;

edgelist elist;

void kruskal();

int find(int belongs[],int vertexno);

void union1(int belongs[],int c1,int c2);

void sort();

void print();

int main()

{

int i,j,total\_cost;

printf("\nEnter number of vertices:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

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

{

for(j = 0;j < n;j ++)

{

scanf("%d",&G[i][j]);

}

}

kruskal();

print();

}

void kruskal()

{

int belongs[MAX];

int i,j,cno1,cno2;

elist.n = 0;

for(i = 1;i < n;i ++)

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

{

if(G[i][j] != 0)

{

elist.data[elist.n].u = i;

elist.data[elist.n].v = j;

elist.data[elist.n].w = G[i][j];

elist.n ++;

}

}

sort();

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

belongs[i] = i;

spanlist.n = 0;

for(i = 0;i < elist.n;i ++)

{

cno1=find(belongs,elist.data[i].u);

cno2=find(belongs,elist.data[i].v);

if(cno1 != cno2)

{

spanlist.data[spanlist.n] = elist.data[i];

spanlist.n = spanlist.n+1;

union1(belongs,cno1,cno2);

}

}

}

int find(int belongs[],int vertexno)

{

return(belongs[vertexno]);

}

void union1(int belongs[],int c1,int c2)

{

int i;

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

if(belongs[i] == c2)

belongs[i] = c1;

}

void sort()

{

int i,j;

edge temp;

for(i = 1;i < elist.n;i ++)

for(j = 0;j < elist.n-1;j ++)

if(elist.data[j].w > elist.data[j+1].w)

{

temp = elist.data[j];

elist.data[j] = elist.data[j+1];

elist.data[j+1] = temp;

}

}

void print()

{

int i;

for(i = 0;i < spanlist.n;i ++)

{

printf("\n%d　 %d　 %d",spanlist.data[i].u,spanlist.data[i].v,spanlist.data[i].w);

}

printf("\n");

}

Input

Enter number of vertices:6

Enter the adjacency matrix:

0 3 1 6 0 0

3 0 5 0 3 0

1 5 0 5 6 4

6 0 5 0 0 2

0 3 6 0 0 6

0 0 4 2 6 0

Output

2　 0　 1

5　 3　 2

1　 0　 3

4　 1　 3

5　 2　 4