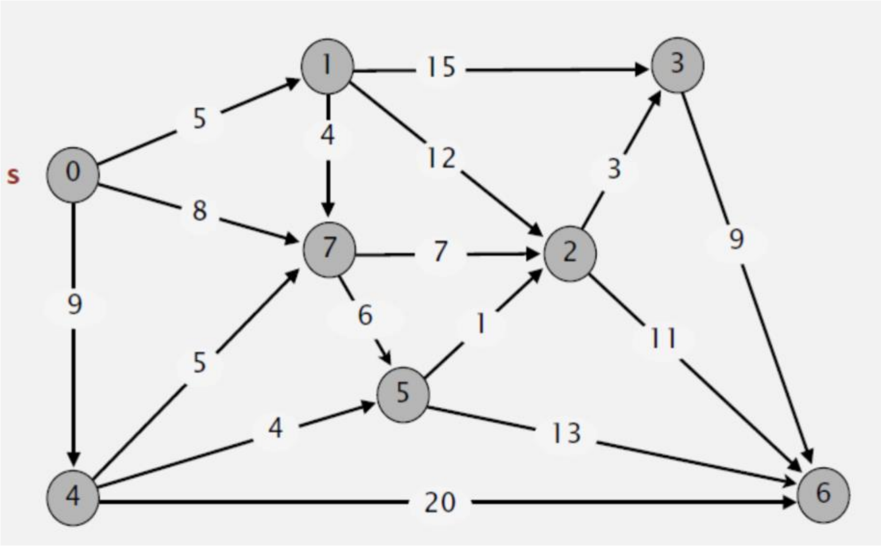
Exercise 5. Answer Sheet

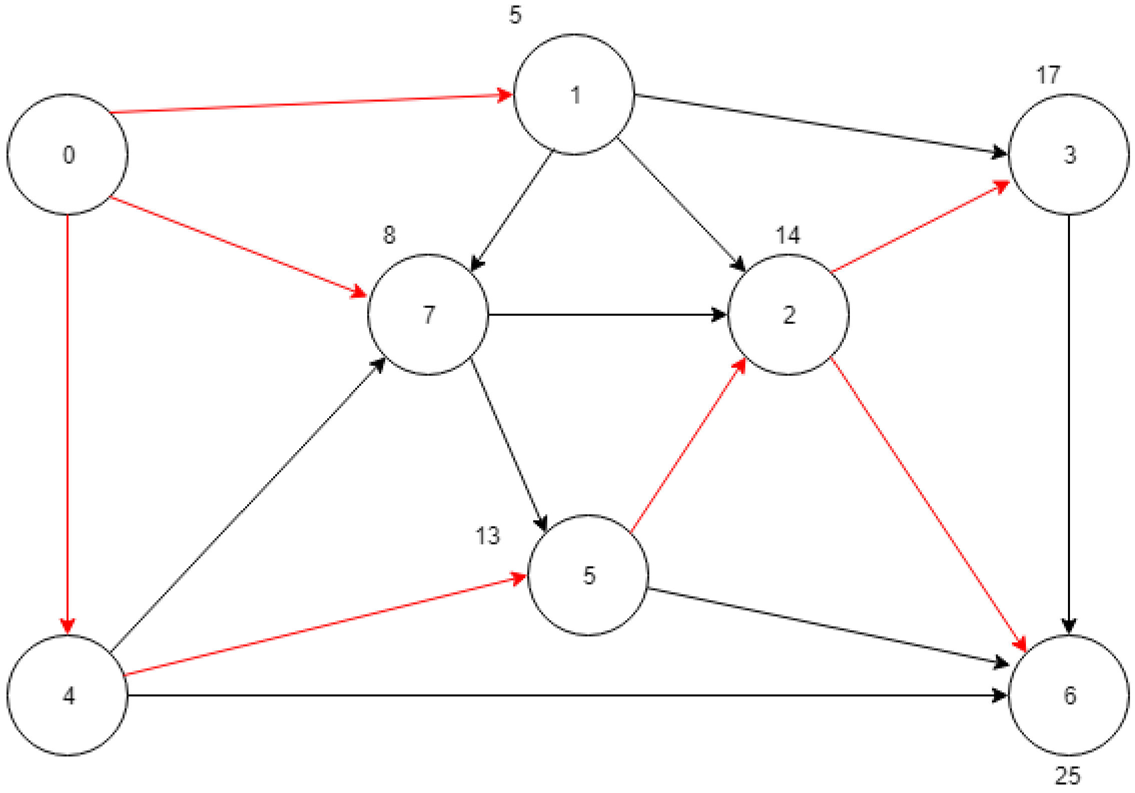
Student's Name:Tsuyoshi kumamoto Student's ID:s1250050

Problem 1. *(15 points)* Consider the graph below.



Draw a shortest path spanning tree with root at vertex ***s***. Show the cost (weight) of paths to each vertex.

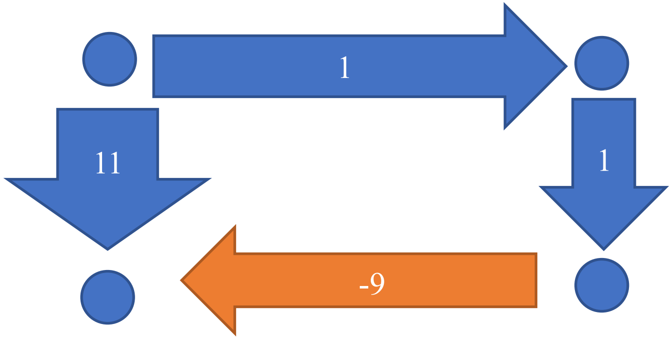
Put your answer here.



Problem 2. *(15 points)* Dijksta's algorithm cannot handle negative weights. Show an example and explain what happens.

Put your answer here.

If there is a negative weight, it will fall into a cycle. Example:

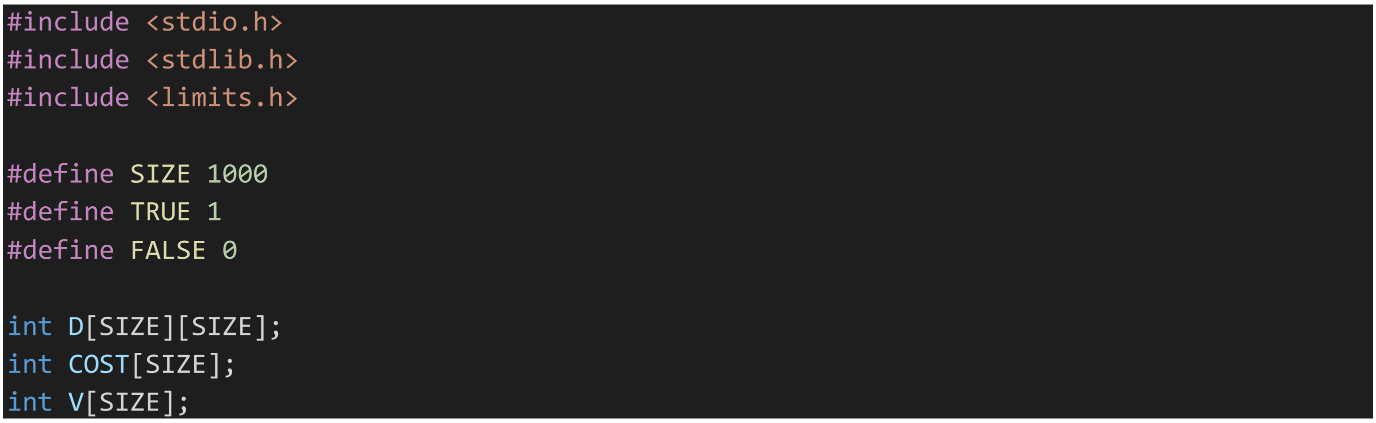


Problem 3. *(20 points)* Extend the pseudocode of the Bellman-Ford algorithm given at the lecture so it can detect negative cycles.

```
def Bellman-Ford-modified (G,s,w):
Init-SS (G,s)
for i=1 to |G.V|-1
    for each edge (u,v) in G.E
              RELAX (u,v,w)
    Add your code here.
            u := uv.source
             v := uv.destination
            if u.distance + uv.weight < v.distance:
                    error message
```

Problem 4. *(50 points)* Write a program implementing Dijksta's algorithm. Upload your source code. Show your input graph and the obtained shortest path spanning tree in the space below.

Put your answer here.



```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define SIZE 1000
#define TRUE 1
#define FALSE 0

int D[SIZE][SIZE];
int COST[SIZE];
int V[SIZE];
int N;
char USE[SIZE];

int dijkstra(int s, int g){
  int min, target;
  int i,neear;
  COST[s] = 0;

  while(1){
    min = INT_MAX;
    for(i=0; i<N; i++){
      if(!USE[i] && min > COST[i]){
  min = COST[i];
  target = i;
      }
    }

    if(target == g){
      return COST[g];
    }

    for(neear = 0; neear<N; neear++){
      if(COST[neear]>D[target][neear] + COST[target]){
  COST[neear] = D[target][neear] + COST[target];
  V[neear] = target;
      }
    }
    USE[target] = TRUE;
  }
}

int main(){
  int r;
  int a,b,l;
  int s,d;

  int i,j,node;
  for(i=0; i<SIZE; i++){
    COST[i] = INT_MAX;
    USE[i] = FALSE;
    V[i] = -1;
    for(j=0; j<SIZE; j++){
      D[i][j] = INT_MAX;
    }
  }

  printf("バーテックスの数を入力:input vertex number\n");
  scanf("%d",&N);
  printf("ルートの数の入力:Root number\n");
  scanf("%d",&r);

  for(i=0; i<r; i++){
    printf("道の両端のバーテックスとその道の距離を入力\n");
    scanf("%d %d %d",&a,&b,&l);
    D[a][b]=l;
  }
  scanf("%d %d",&s,&d);

  printf("距離:%d\n",dijkstra(s,d));

  node = d;
  printf("%d",node);
  while(1){
    node = V[node];
    printf(" -> %d",node);
    if(node == s){
      break;
    }
  }

  return 0;
}
```

The correction was not in time.