University of Nevada, Las Vegas Computer Science 477/677 Spring 2025

Assignment 4: Due Saturday March 1, 2025

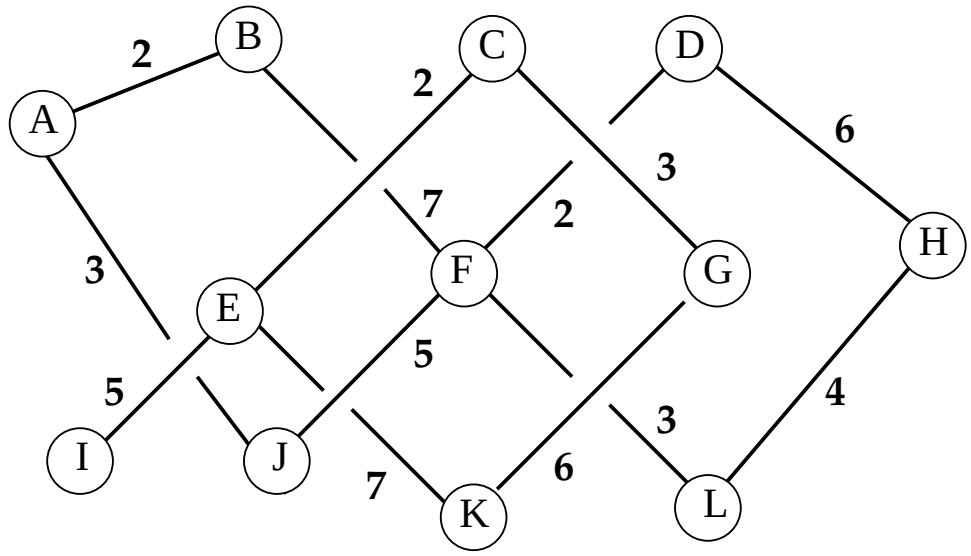
Name:_____

You are permitted to work in groups, get help from others, read books, and use the internet.
To turn in the homework, follow instructions given by the teaching assistant, Sabrina Wallace.

1. Find an optimal Huffman code on the alphabet A,B,C,D,E,F where frequencies are given in the following table.

A	8
B	6
C	3
D	11
E	14
F	2

2. The weighted graph below has more than one component. Use Kruskal's algorithm to construct a subgraph which consists of a minimal spanning tree of each component.



3. Walk through heapsort, using the method which has a *heapify* phase, for the array: **A Q R B X S M L N T** To make the problem easier to grade, please show the evolution of the array in the table below. If more rows are needed, draw them in. Indicate the step at which heapify has been completed.

1	2	3	4	5	6	7	8	9	10
A	Q	R	B	X	S	M	L	N	T

4. Here is C++ code for quicksort. Assume that the global array `int A[n]` has been declared. The goal is to sort A.

```
int const n = 20;
int lo;
int hi;

int A[n];

void swap(int&x, int&y)
 {
  int z = x;
  x = y;
  y = z;
 }

void display()
 {
  for(int i = 0; i < n; i++)
  cout << " " << A[i];
  cout << endl;
 }

void quicksort(int first, int last)
 // sorts A[first..last]
 {
  assert(0 <= first and first <= last and last < n);
  if(first < last) // What happens if first = last?
   {
    int pivot;
    int mid = (first+last)/2;
    pivot = A[first];
    swap(A[first],A[mid]);
    pivot = A[first];
    lo = first;
    hi = last;
    while(lo < hi) // This is the main loop
     {
      if(A[hi] < A[lo+1])
       swap(A[hi],A[lo+1]);
      while(A[lo+1] < pivot) lo++;
      while(A[hi] > pivot) hi--;
     }
```

```
    assert(lo == hi);
    swap(A[first],A[lo]);
    quicksort(first,lo);
    quicksort(lo+1,last); // Oops!  There might be a bug here!
   }
 }

int main()
 {
  display();
  quicksort(0,n-1);
  display();
  return 1;
 }
```

The output of my program:

```
70 29 18 45 91 58 64 97 95 67 92 19 63 27 84 73 60 72 42 99
18 19 27 29 42 45 58 60 63 64 67 70 72 73 84 91 92 95 97 99
```

Answer the following questions.

(a) What is the loop invariant of the main loop?

(b) I was careful not to have any duplicate entries in my initial array, the first line of my output. Did I really need that condition for the program to work correctly? Explain.

(c) I treated the case where the array has length 2 with special code. Was that necessary?

(d) There might be a bug in the program, as indicated at the second recursive call. Can you find it? Even if there is a bug, the program might run correctly, but it might not.