CS 303 Algorithms and Data Structures
Final Exam

105 points. 150 minutes. **Closed book/notes.**

Name: ______________________________

1. Illustrate the operation of sorting the sequence **52, 23, 46, 99,121, 256, 68, 47** using **heapsort**. **[10 points]**

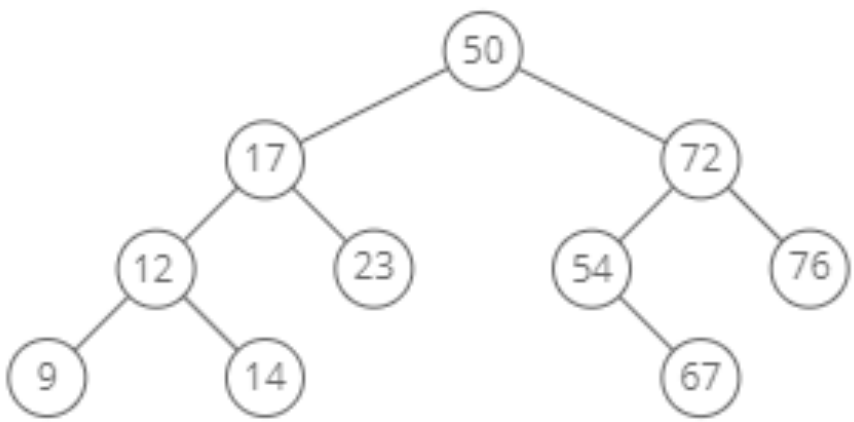
   The heapsort algorithm is given below.

```
HEAPSORT(A)
BUILD-MAX-HEAP(A)
for i = A.length downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

```
BUILD-MAX-HEAP(A)
A.heap-size = A.length
for i = ⌊A.length/2⌋ downto 1
    MAX-HEAPIFY(A, i)
```

```
MAX-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heap-size and A[l] > A[i]
    largest = l
else largest = i
if r ≤ A.heap-size and A[r] > A[largest]
    largest = r
if largest ≠ i
    exchange A[i] with A[largest]
    MAX-HEAPIFY(A, largest)
```

2. Show what the tree would look like after the following changes are made to the following binary search tree (make sure to include all the intermediate steps and explain each step): **[12 points]**
   a) **Delete** node with value 17
   b) **Insert** node with value 56
   c) **Insert** node with value 88
   d) **Delete** node with value 72



The algorithm for deleting a node in a binary search tree is given below:

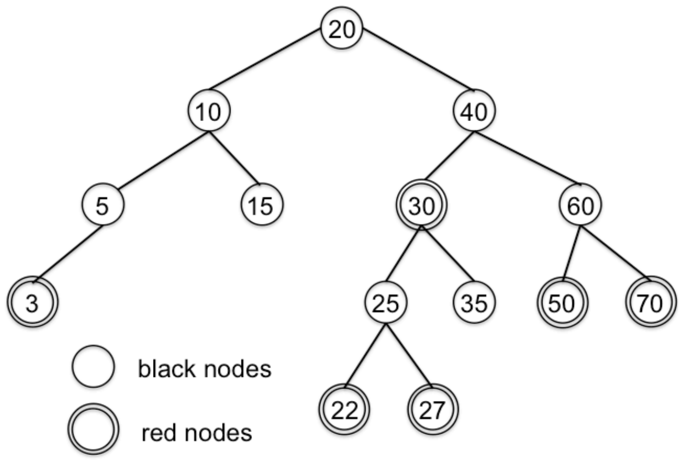
```
TREE-DELETE(T, z)
  if z.left == NIL
      TRANSPLANT(T, z, z.right)          // z has no left child
  elseif z.right == NIL
      TRANSPLANT(T, z, z.left)           // z has just a left child
  else // z has two children.
      y = TREE-MINIMUM(z.right)          // y is z's successor
      if y.p ≠ z
          // y lies within z's right subtree but is not the root of this subtree.
          TRANSPLANT(T, y, y.right)
          y.right = z.right
          y.right.p = y
      // Replace z by y.
      TRANSPLANT(T, z, y)
      y.left = z.left
      y.left.p = y
```

```
TRANSPLANT(T, u, v)
  if u.p == NIL
      T.root = v
  elseif u == u.p.left
      u.p.left = v
  else u.p.right = v
  if v ≠ NIL
      v.p = u.p
```

3. Is the following tree a red-black tree? Explain your answer. If not, make appropriate changes to make this a valid red-black tree. [**2 points**]

   Show what the tree would look like after you insert the following values to the valid red-black tree obtained from above (make sure to explain all the steps involved and mark the red and black nodes accordingly): **[10 points]**

   a. 17
   b. 76
   c. 2



The pseudo-code for red-black tree insert is given below.

```
RB-INSERT(T, z)
  y = T.nil
  x = T.root
  while x ≠ T.nil
      y = x
      if z.key < x.key
          x = x.left
      else x = x.right
  z.p = y
  if y == T.nil
      T.root = z
  elseif z.key < y.key
      y.left = z
  else y.right = z
  z.left = T.nil
  z.right = T.nil
  z.color = RED
  RB-INSERT-FIXUP(T, z)
```

```
RB-INSERT-FIXUP(T, z)
  while z.p.color == RED
      if z.p == z.p.p.left
          y = z.p.p.right
          if y.color == RED
              z.p.color = BLACK
              y.color = BLACK
              z.p.p.color = RED
              z = z.p.p
          else if z == z.p.right
                  z = z.p
                  LEFT-ROTATE(T, z)
              z.p.color = BLACK
              z.p.p.color = RED
              RIGHT-ROTATE(T, z.p.p)
      else (same as then clause with "right" and "left" exchanged)
  T.root.color = BLACK
```

```
LEFT-ROTATE(T, x)
  y = x.right            // set y
  x.right = y.left       // turn y's left subtree into x's right subtree
  if y.left ≠ T.nil
      y.left.p = x
  y.p = x.p              // link x's parent to y
  if x.p == T.nil
      T.root = y
  elseif x == x.p.left
      x.p.left = y
  else x.p.right = y
  y.left = x             // put x on y's left
  x.p = y
```

4. For the given directed, edge-weighted input graph:

D	C	10
A	B	10
D	A	12
A	C	15
C	D	30
B	C	20
C	F	45
D	E	35
E	F	25

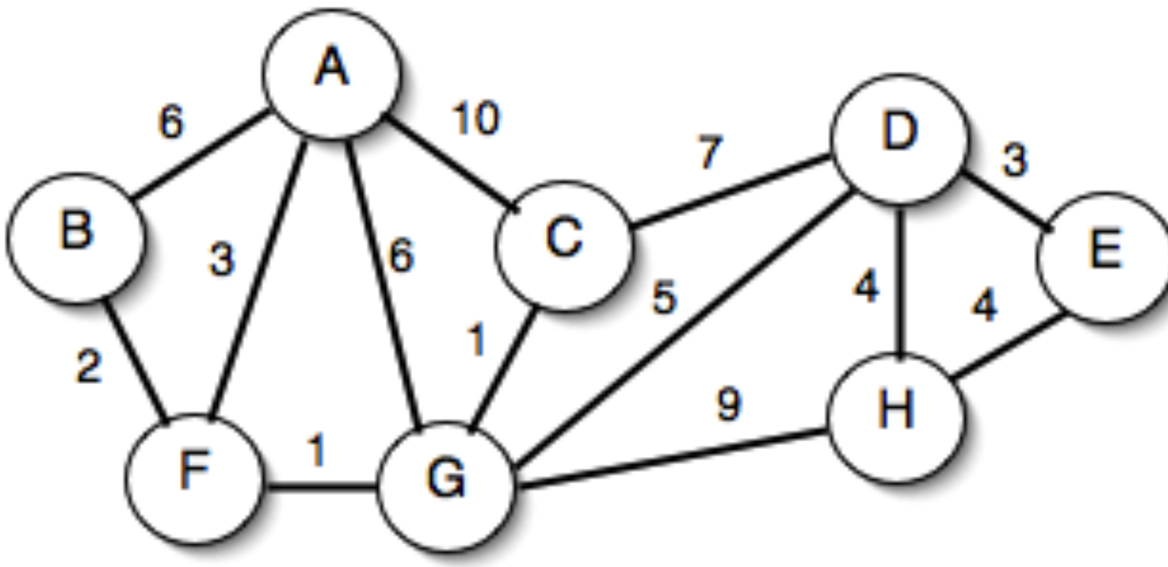
   a. Draw the graph that corresponds to the given input. **[7 points]**

   b. Draw the adjacency matrix representation for this graph. **[5 points]**

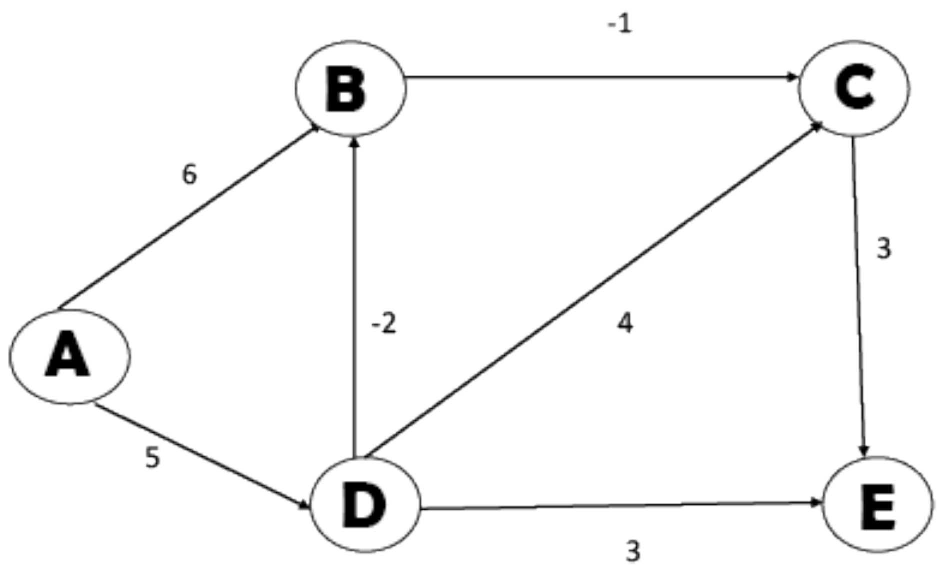
c. Draw the adjacency list representation for this graph. **[5 points]**

d. Which algorithm would you use to compute the shortest path between node A and all other nodes in the graph. Use that algorithm to compute the shortest path from node A to all other nodes (show the steps involved in computing the shortest paths) and write down the sequence of nodes traversed by the shortest paths. **[10 points]**

5. For the given edge-weighted graph compute the **minimum spanning tree**. Show the intermediate steps involved in computing the minimum spanning tree. Explain which algorithm you are using. **[10 points]**



6. Which algorithm would you use to compute the shortest path between node A and all other nodes in the graph. Use that algorithm to compute the shortest path from node A to all other nodes (show the steps involved in computing the shortest paths). **[10 points]**



7. Explain the differences between Prim's and Kruskal's Algorithms **[8 points]**

8. Explain the differences between Dijkstra's and Bellman Ford's Algorithms **[8 points]**

9. Explain the Dynamic Programming approach and give an example that you prefer to use Dynamic Programming Approach to solve **[8 points]**