Data Structures

Time: 3 Hours | Final Exam – June 28 | Total Marks: 50

Note: You are not required to write any code for questions Q1-Q5. Use of calculator/mobile is not allowed.

Q1: Given the following function, what will be the output of `A(1)`? {4}

```
int A(int n) {
  if (n > 5)
    return n;
  else
    return n+2*A(n*2)+3*A(n*3);
}
```

Q2: How an empty AVL tree will look like after inserting the following values in the given order: 3, 1, 2, 5, 4, 6? {4}

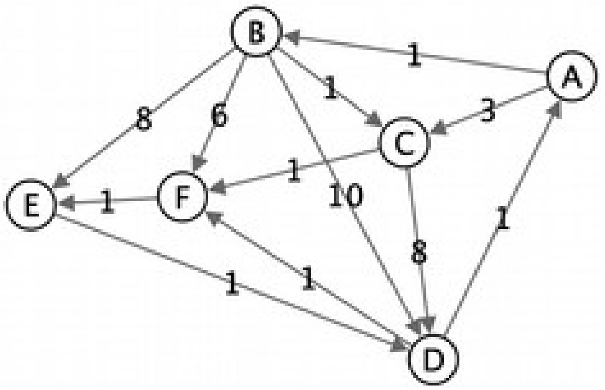
Q3: Given the following hash functions: {4}

```
int h1(int k, int m) {
    return k % m;
}

int h2(int k, int m) {
    return 1 + (k % (m-6));
}
```

Given *m*=11, how a hash table of size *m* will look like after inserting the following values: 5, 27, 30, 31, 13, 23, 38, 46. Use double hashing for insertions. *h1* is used for initial probing and in case there is a collision, *h2* is used.

Q4: Given the following graph: {4+2=6}



a. Find the shortest path from vertex A to all the other vertices using *Dijkstra* algorithm.
b. Show the adjacency matrix for this graph.

Q5: Sort the given values using the following algorithms. Show all the intermediate steps. {2+4=6}
a. Using heap sort: 1, 4, 3, 6, 2, 5
b. Using counting sort: 5, 4, 2, 1, 2, 4, 2

Q6: Implement the *insert* function for the *sorted circular linked list* ADT. Please recall that, in circular linked list, only a single pointer to the list is maintained and the last node of the list is connected to the first node. {6}

Q7: Implement the *delete* function of the binary search tree. {4}

Q8: Implement an iterative function `bool CheckMinHeap(int A[], int first, int last)`, which takes an array and indexes of its first and last element and returns true if the array contains a min-heap structure, false otherwise. The function should not take more than *n/2* steps. {6}

Q9: Implement the *delete* function using quadratic probing for open addressing based hash table. {4}

Q10: Implement a function *Degree*, which displays the degrees of all the nodes in a graph. Given that the graph is undirected and the graph ADT is implemented using an adjacency matrix. {6}