CSCI 260 Test 4 Dec 4 2017 Name: ____________________________________

1. [5 marks] Recall that the efficient implementation of MakeHeap sends elements in the heap downwards to achieve heap order on the keys. In the context of **Min Heaps**, suppose MakeHeap is called on the following array. Show the state of the **array** after MakeHeap has been executed on it.

   HeapArray=[12,100,8,11,9,4,7].

2. [5 marks] Suppose you perform HeapInsert(4) on a heap whose contents are form the following heap-array:
   HeapArray = [5,10,6,11,14,8,13,17,12,15,80,9]
   Give the heap-array or the heap tree that results.

3. [5 marks] Show the results of performing a RemoveMin on the following heap by giving either the heap-array or heap-tree that results.
   HeapArray = [5,9,8,11,10,15,19,12,13]

4. Let coin denominations be d1=8, d2=5, and d3=1.
   a) [1 mark] How many coins and which ones would the Greedy Algorithms decide should be used to make 15 cents?
   b) [4 marks] Use Dynamic Programming to determine the optimal number of coins to make 15 cents. Show the entire table, call it C, from 0 up to and including 15. The table entries should consist of two fields; one should say how many coins are needed, and the other says what the largest denomination is in the optimal coinset for that amount.
   c) [4 marks] In the above Dynamic Programming solution, let C[i].num denote the number of coins, and C[i].maxdenom be the largest denomination coin in the optimizing coinset. Give an algorithm PrintChange(int x) to output the coin denominations to make up a given amount that is within the range of the table. You can assume that the table is already populated with values. You can use cout to put the coin denomination to the output. For example, the following call might generate the following output:

```
>> PrintChange(7)
coin of denomination 5
coin of denomination 1
coin of denomination 1
```

5. [8 marks] Below are two copies of the same graph. Run Prim's on the one to the left, bolding each edge that you include in the tree, and writing beside the weight the number of the edge in the order you added it – i.e., write a 1 beside the first edge you add, 2 beside the second one, etc – parentheses have been provided for this purpose. For Prim's, start at v0. Run Kruskal's on the right graph, labelling the edges with the order that you added them, as in Prim's.

v0: (v1,2) (v3,12) (v4,6)
v1: (v0,2) (v2,7) (v4,5)
v2: (v1,7) (v4,8) (v5,13)
v3: (v0,12) (v4,11) (v6,4)
v4: (v0,6) (v1,5) (v2,8) (v3,11) (v5,12)
v5: (v2,13) (v4,12) (v6,9) (v7,3) (v8,14)
v6: (v3,4) (v5,9) (v7,16)
v7: (v3,10) (v5,3) (v6,16) (v8,15)
v8: (v5,14) (v7,15)

6. [10 marks] Run Dijkstra's algorithm on the following graph. You are to populate the table given with values. The values are to be the "temporary" distance values that are computed for each vertex (one vertex per row), across all the iterations of the Relaxation loop (one iteration per column). Once a "temporary" value becomes "permanent", underline it. The start vertex is s. Head each column with the vertex whose label was made permanent in the previous iteration. The first column has been done for you.

Graph: (given as adjacency lists)

s: (a,9) (c,15) (e,10)
a: (s,9) (b,1) (d,7) (e,8)
b: (a,1) (c,7) (d,3) (e,4)
c: (s,15) (b,7) (d,2)
d: (a,7) (b,3) (c,2) (e,11)
e: (s,20) (a,8) (b,4) (d,11)

Vertex		s							
s	0								
a	infinity	9							
b	infinity	infinity							
c	infinity	15							
d	infinity	infinity							
e	infinity	20							

Give all the lengths of the shortest paths from s to each vertex in the graph.