Name_____ ID #_____ Seat No. _____

King Mongkut's University of Technology Thonburi
Final Examination of Second Semester, Academic Year 2015

COURSE CPE 113 Algorithms and Data Structures **Automation Engineering, 1st Yr.**
Friday, May 13, 2016 **9.00-12.00 h.**

Instructions

1. This examination contains 11 questions, 10 pages (including this cover page).
2. The answers must be written in the examination paper. You may use the back of the paper.
3. No books, notes, calculators or any other documents can be taken into the examination room.
4. Use your consideration and explain it if you have certain doubts about the exam questions.

Students must raise their hand to inform to the proctor upon their completion of the examination, to ask for permission to leave the examination room.
Students must not take the examination and the answers out of the examination room.

Students will be punished if they violate any examination rules. The highest punishment is dismissal.

This examination is created by

Asst. Prof. Nuttanart Facundes, Ph.D.

Tel. 0-2470-9256

Name_____ ID #_____

Total points = 40 points (25%)

Sorting Algorithms (5 points)

1. Illustrate the operation of **straight insertion sort** by completing the left table below. In successive rows of the table, show the array contents after each pass of the algorithm.

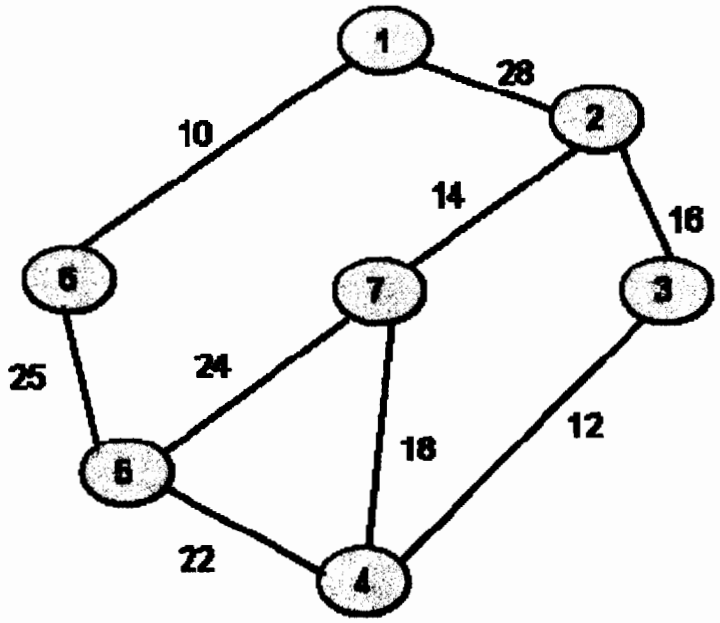
2. Illustrate the operation of **bubble sort** by completing the right table below. In successive rows of the table, show the array contents after each pass of the algorithm.

9	8	6	7	5	0

9	8	6	7	5	0

Name_____ ID #_____

3. For the following graph, draw an adjacency list representation (5 points)



Name_____ ID #_____

Hashing (5 points)

4. Consider a hash table with ten slots. For the hash function h(x) = x mod 10, insert the keys (33, 54, 69, 74, 18, 19) in the order given into the table. Use *linear probing* for solving collisions. Show the results in the table below.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

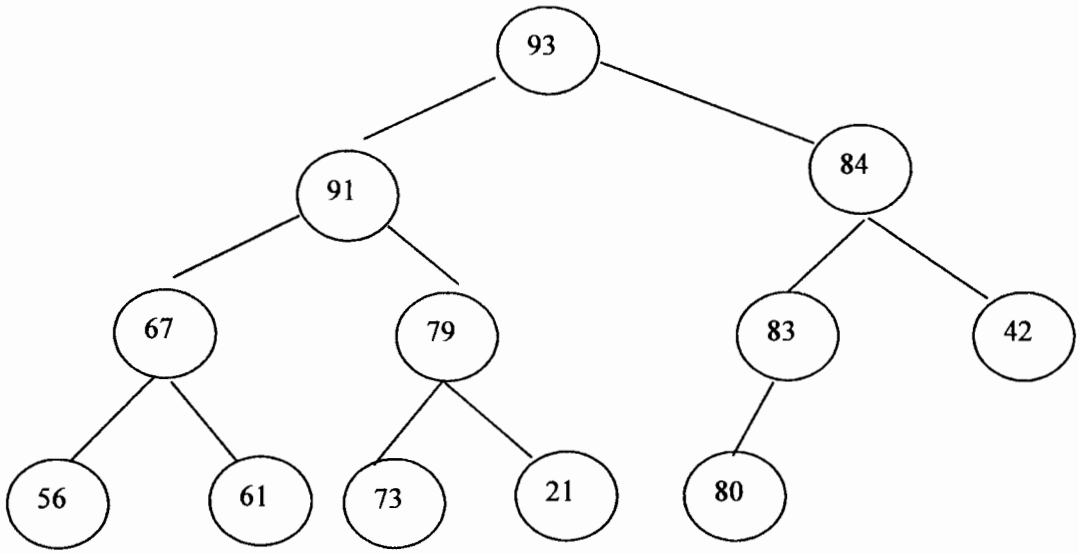
5. Consider a hash table with ten slots. For the hash function h(x) = x mod 10, insert the keys (33, 54, 69, 74, 18, 19) in the order given into the table. Use *quadratic probing* for solving collisions. Show the results in the table below.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Name_____ ID #_____

6. Heap (5 points)

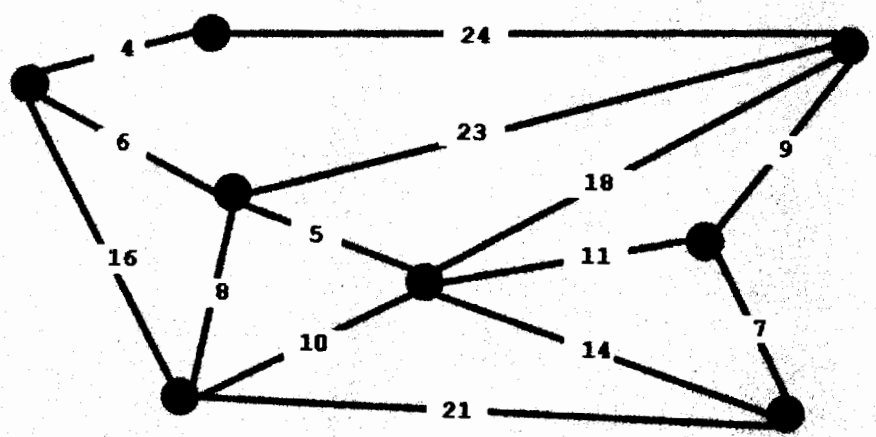
6.1 Consider the heap given below. If the heap elements were stored in an array, what element would be stored at index 7?



6.2 Based on the heap above, show the heap sort for the first 4 passes in the following table and explain how you do the heap sort.

Name_____ ID #_____

7. Find the minimum spanning tree of the following graph (5 points)



Name_____ ID #_____

8. Huffman Coding (5 points)

8.1 Given letters A,B,C,D,E,F,G with respective frequencies: 0.1 (A), 0.22 (B), 0.23 (C), 0.11 (D), 0.02 (E), 0.03 (F), 0.29 (G). Build the Huffman code for these letters by drawing the Huffman tree.

8.2 Huffman coding belongs to which type of algorithm?

Name_____ ID #_____

9. Given an array of numbers below, search for 146 using binary search. Show the details of *first*, *last* and *mid* in each loop. (5 points)

1,3,4,6,7,9,10,12,13,15,16,145,147,148,150

Name_____ ID #_____

From lab assignments (5 points)

10. In one lab, we have learnt about Heap Sort, the algorithms for sorting the number in array by ascending order. Normally, the numbers in array to be sorted are in random order. The first thing to work with the array is to **heapify** it or turn it into a heap. Then, the algorithm iteratively swap the number in the first element to the back and then reheapDown the remaining numbers.

For reheapUp, your heap has n-1 items and the 1 newly added item at the last of array. The reheapUp is recursively swap the last item with its parent if its data is higher than its parent's data.

For reheapDown, the new item is at root of the array. The root has to select one of children with highest value to be swapped with and recursively do it whenever there is no higher child's data.

Given the function declaration of reheapUp and reheapDown as follow

```
void reheapUp(int* arr, int childLoc);
void reheapDown(int* arr, int arrSize, int rootLoc);
```

Question: What is the correct heapify algorithm? Choose from below:

(A)

```
void heapify(int* arr, int arrSize){
    int worker;
    for(worker = 0; worker < arrSize; worker++){
        reheapUp(arr, worker);
    }
}
```

(B)

```
void heapify(int* arr, int arrSize){
    int worker;
    for(worker = 0; worker < arrSize; worker++){
        reheapDown(arr, worker);
    }
}
```

(C)

```
void heapify(int* arr, int arrSize){
    int worker;
    for(worker = 0; worker < arrSize; worker++){
        reheapUp(arr, worker, 0);
    }
}
```

(D)

```
void heapify(int* arr, int arrSize){
    int worker;
    for(worker = 0; worker < arrSize; worker++){
        reheapDown(arr, worker, 0);
    }
}
```

Name_____ ID #_____

11. In one lab, we have learnt about graph data structure which was implemented by the array of integer as nodes or vertices and the adjacency matrix as the edges or connection between nodes.

This following code is used in the **delete node** function.

```
for(i=removingIndex; i<graph->node_count - 1; i++)
{
    for(j=0; j<graph->node_count; j++)
    {
        graph->adjacency[i][j] = graph->adjacency[i+1][j];
    }
}
for(i=removingIndex; i<graph->node_count - 1; i++)
{
    for(j=0; j<graph->node_count; j++)
    {
        graph->adjacency[j][i] = graph->adjacency[j][i+1];
    }
}
```

The variable called removingIndex is the array index in which the node is being deleted.

What does the above code do exactly? Explain briefly