CENG 213 Sample Midterm Questions

Q1. a) What is the running time complexity of the following code in Big O notation?

```
i=1;
sum = 0;
while (i <= n) {
   j = i;
   while (j <= n) {
      sum = sum + i;
      j = j + 1;
   }
   i = i + 1;
}
```

b) What is the running time complexity of the following code in Big O notation?

```
int Fact(n) {
   if (n<3)
       return n;

   return n * (n-1) * Fact(n-2);
}
```

c) What is the running time complexity of the following recursive function in Big O notation?

```
int foo(n) {
   if (n > 0)
   {
       return 1 + foo(n/2) + foo(n/2);
   }
   else {
       return 0;
   }
}
```

d) What will be the return value of the above function if called with `foo(8)`?

Q2. What are the Big-O time complexities of the following operations under the given assumptions. You can assume that N represents the number of elements.

a) Selection sort on an already sorted array of integers

b) Insertion sort on an already sorted array of integers

c) Bubble sort (without "isSorted" flag optimization) on an already sorted array of integers

d) An unsuccessful sequential search on an unsorted array of integers

e) An unsuccessful binary search on a sorted array of integers

f) A successful sequential search on a sorted vector of integers (assume average case)

g) Searching for the last element in a circular doubly-linked list

Q3. Consider the following list node structure:

```
template <class ListDataType>
struct Node {
   ListDataType item; // the data of the node
   Node* next; // points to the next node of the list
   Node* prev; // points to the previous node of the list
};
```

Assume that we have a doubly linked list with a dummy head node pointed by pointer `Head` and at least one other internal node pointed by pointer M, which is not the last node. Write few lines of code to accomplish the following. <u>You may NOT swap data to accomplish any of the following operations</u>. For each operation, assume the original list is described as above. You are encouraged to draw pictures to justify your code. Note that for each operation, you need to manipulate at least two pointers, `next` and `prev`.

a) Delete the first node.

b) Insert a node P immediately after M.

c) Make the node M the first one in the list.

d) Make M point to the i^{th} node. Make it NULL if i is out of range of the list.

Q4. Implement the queue operations **enqueue** and **dequeue** by using **only two stacks** and **no other extra variables**. Use the stack interface shown on the right.

```
template <class T>
class Queue {
public:
// inserts the item to the queue
    void enqueue(const T& item) {

}
// removes and returns an item from the
queue
    T dequeue() {

 }

private:
    Stack<T> s1;
    Stack<T> s2;
};
```

```
template <class T>
class Stack {
public:
// default constructor
    Stack();

// pushes the item to the stack
    void push(const T& item);

// pops and returns an item
from the stack
    T pop();

// returns an item from the
stack
    T top() const;

// returns true if the stack is
empty, false otherwise
    bool isEmpty() const;

private:
// not shown: you do not need
this info
};
```

Q5.

i. In the following questions about binary trees, the height of a tree is the length (number of edges) of the longest path from the root to a leaf. A tree consisting of just one node has height 0.

a) What is the minimum number of leaves in a binary tree of height k? ________________

b) What is the maximum number of leaves in a binary tree of height k? ________________

c) What is the number of internal nodes (non-leaf nodes) in a full binary tree? _______________

ii. a) Draw the *binary search tree* created by inserting these values in this order:
6 8 2 4 3 0 1 9 5 7

b) Give a pre-order traversal of your tree: _______________________________

c) Give a post-order traversal of your tree: _______________________________

d) Delete the root of your tree. Draw the new tree here:

iii. Construct the *Binary Search Tree* using the preorder traversal given as: **E C A B D H F G I J**.