Mock Exam 1

1. Implement the **Node* reverseList(Node* head)** function. Reversing a linked list means that the tail becomes the head, and all the links between the nodes are reversed. Given the head of a singly linked list, reverse the list, and return the reversed list.

Node* reverseList(Node* head){

}

2. Write the function **Node* middleNode(Node* head),** which returns the middle node of a singly linked list. If there are two middle nodes, return the second middle node.

Node* middleNode(Node* head){

}

3. Define the function **Node* detectCycle(Node* head).** Given a circularly linked list, write a function that returns the node at the start of the cycle. If there is no cycle return **nullptr. (HINT: Use a fast and slow pointer)**

Node* detectCycle(Node* head){

}

3. Define the function **bool isPowerOfTwo(int n)**, which returns true if **n** is a power of two, otherwise false. Do this **recursively!**

bool isPowerOfTwo(int n){

}

4. Given a string **s** , write the recursive function **bool isPalindrome(string s)** to check if the string is a palindrome. A string is a **palindrome** if it reads the same forward and backward.

bool isPalindrome(string s){

}

4. Given the following array **[5 3 8 4 2]** show the array after each iteration of **bubble sort**.

[5 3 8 4 2]

5. Given the following array **[10 3 15 7 8]** show the array after each iteration of **selection sort**.

[10 3 15 7 8]

6. What is the time complexity for the code below?

```
for(int i = 0; i < 50; i++)
        cout << i << endl;
```

7. What is the time complexity for the code below?

```
for(int i = 0; i < n; i++){
   for(int j = 0; j < n; j++){
        cout << j << endl;
    }
}
```

8. What is the time complexity for the code below?

```
for(int i = 0; i < n; i *= 2){
   for(int j = 0; j < n; j++){
        cout << j << endl;
    }
}
```