Santa Monica College
Spring 2018

CS 20A: Data Structures with C++

Assignment 4

Name: Nashir Jamohamed **ID:** 1325605

Problem 1:

A palindrome is a word, phrase or sequence that reads the same backward as forwards, for example, "bob", "step on no pets". Write a recursive function isPalindrome that takes a string as input and returns true if it is a palindrome, false otherwise. You might find the string::substr useful, which takes two arguments, the first being the start position of the character in the original string to be copied, the second is the length of the substring to be copied, returns the substring. For example:

```
cout << string("Hello World").substr(2, 5); // Prints "llo W"
```

```
bool isPalindrome(string s) {
    int len = (int)s.length();
    if (len <= 1)
        return true;
    if (s[0] == s[len-1])
        return isPalindrome(s.substr(1, len-2));
    else
        return false;
}
```

Problem 2:

What does the following function compute? 3, 2

```
// Precondition: b is a nonnegative integer
int mystery1(int a, int b) {

        if (b == 0) return 1;

        if (b % 2 == 0) return mystery1(a*a, b / 2);

        return mystery1(a*a, b / 2) * a;
}
```

it computes a^b, or "a" to the power of "b"

Name: Nashir Janmohamed **ID:** 1375605

Problem 3:

What does the following function compute?

```
// Precondition: a and b are nonnegative integers
int mystery2(int a, int b) {
                 3     2
        if (b == 0) return 0;

                                                3  3   1
        if (b % 2 == 0) return mystery2(a + a, b / 2); //returns 6 from here

                                    6  6   0      6
        return mystery2(a + a, b / 2) + a;
}
```

it computes a x b, or "a" multiplied by "b"

Problem 4:

Write a recursive function printReverse that takes an array of integers and its size as inputs, and prints its elements in reverse order. For example, if we pass into this function the array 1, 4, 3, 6 and the size of 4, we should see 6, 3, 4, 1 in the console.

```
void printReverse(int *arr, int size){
    if (size == 0)
        return;

    cout << arr[size-1];

    printReverse(arr, size-1);

}
```

Name: Nasher Jammohamed **ID:** 1325605

Problem 5:

Implement a recursive function sumOfDigits that takes an positive integer as input and returns the sum of all of the digits in the integer.

```
int sumOfDigits(int n){

    if (n < 10)
        return n;
    else
        return sumOfDigits(n/10) + (n % 10);

}
```

Problem 6

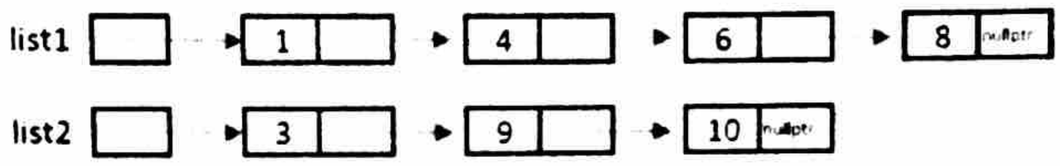
Implement a recursive function deleteList, that takes pointer to the head of a singly linked list and deletes the whole list.

```
struct Node {
        int val;
        Node* next;
};

void deleteList(Node* head) {
        if (head == nullptr)
            return;

        deleteList(head->next);
        delete head;
}
```

Name: Nashir Jamohamed **ID:** 1325605

Problem 7:

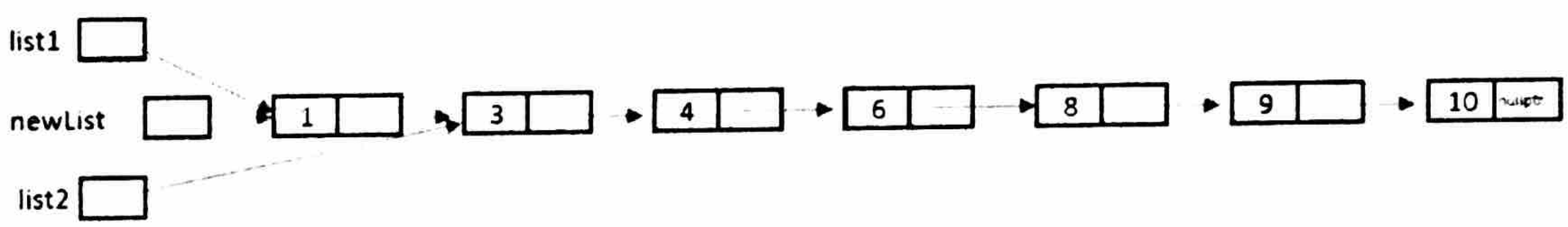
Given the same Node above, implement a recursive function that merges two sorted singly linked lists into a single sorted linked list. The function should return the head of the new list. You may not create any new Nodes, this is known as an in-place merge. For example suppose we have two list:



And we call our merge function on these two lists:

```
Node* newList = inPlaceMerge(list1, list2);
```

The state of our program after that function call may look like:




```
Node* inPlaceMerge(Node* list1, Node* list2) {

if (list2 == nullptr)
    return list1;
else if (list1 == nullptr)
    return list2;
else {
    Node* newList;
    if (list1->val < list2->val) {
        newList = list1;
        newList->next = inPlaceMerge(list1->next, list2);
    }else{
        newList = list2;
        newList->next = inPlaceMerge(list1, list2->next);
    }
    return newList;
}

}
```