Homework 6: Recursive Algorithms Review

Instructor: Mehmet Emre

CS 32 Spring '22

Due: 4/20 12:30pm
Name & Perm # (no partners allowed): Bharat Kathi (5938444)

Reading: DS 9.1, PS 14.1 and 14.2

1

List two important pieces of information stored in an **activation record** (as discussed on p. 442 in DS):

1. (5 pts)

   Answer: where the function should return when it's done with its computation

2. (5 pts)

   Answer: values of the functions local variables and parameters

2

There are two important parts to every simple recursive function: the base case, and the recursive call that makes progress towards the base case.[1] Something that can go wrong with recursion when it is used incorrectly is a **stack overflow**. Explain two different ways that a recursive function could be writen incorrectly that could lead to stack overflow. Hint: one has something to do with the base case, and the other with the recursive call.

1. (5 pts)

   Answer: If you have an incorrectly setup base case (or none at all) the function will never stop calling itself, resulting in a stack overflow.

2. (5 pts)

   Answer: You can also just have too many recursive calls before the base case is hit, leading to a stack overflow.

[1] There are other forms of recursion, such as "mutual recursion", where `foo()` calls `bar()` and `bar()` calls `foo()`, but let's set those aside for the moment, and deal only with simple recursive functions.

3

Given a fairly common definition for a **struct Node** that can be used to make a singly linked list of int values:

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

1. (10pts) Write an **iterative** function `printList(const Node* head)` that takes a pointer to the head Node of the singly linked list and prints each value of the linked list, one per line. Write the entire function (including the function signature), and be sure to write correct and compilable C++ code in your solution.

   ```
   #include <iostream>
   using namespace std;

   void printList(const Node* head) const {
     Node *n = head;
     while (n) {
       cout << n->data << endl;
       n = n->next;
     }
   }
   ```

2. (10 pts) Rewrite the `printList(const Node* head)` function from part 1 **recursively**.

   ```
   #include <iostream>
   using namespace std;

   void printList(const Node* head) const {
     if (head == nullptr) return;
     cout << head->data << endl;
     printList(head->next);
   }
   ```