CMPT 125, Fall 2021

Final Exam
December 17, 2021

Name________________________

SFU ID: |__|__|__|__|__|__|__|__|__|

Instructions:
1. Duration of the exam is 180 minutes.
2. Write your name and SFU ID **clearly**.
3. This is a closed book exam, no calculators, cell phones, or any other material.
4. The exam consists of four (4) problems. Each problem is worth 25 points.
5. Write your answers in the provided space.
6. There is an extra page at the end of the exam. You may use it if needed.
7. Explain all your answers.
8. Really, explain all your answers.

Good luck!

Problem 1 [25 points]

a) The function below gets as input an array of ints of length n..

```
void what(int* ar, int n) {
     if (n>1) {
         what(ar, n-1);
         char tmp = ar[n-2];
         ar[n-2] = ar[n-1];
         ar[n-1] = tmp;
     }
}
```

[6 points] What is the time complexity of what() as a function of n?. Use Big-O notation to express your answer. Explain your solution.

[6 points] What is the effect of what() when applied on the array [1,2,3,4,5]?
What is the functionality of what()? Explain your answer.

b) [7 points] What will be the output of the C++ program below? Explain your answer.

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

class Test {
  private:
    int m_x;

  public:
    Test() {
      m_x=0;
    }
    Test(int x) {
      m_x=x;
    }
    Test(Test& other) {
      this->m_x = other.m_x+1;
    }
    int getX() { return m_x; }
    void setX(int x) { m_x=x; }
};

int main() {
  Test t;
  cout << "t = " << t.getX() << endl;
  t.setX(4);
  cout << "t = " << t.getX() << endl;
  Test s(t);
  cout << "s = " << s.getX() << endl;
  return 0;
}
```

c) [6 points] Explain the difference between pointers and references in C++. Provide an example if needed.

Problem 2 [25 points]

A ***Doubly Linked List*** is a linked list where each node has a pointer to the next element and to the previous element.

```
struct DLL_node {
  int value;
  struct DLL_node* next;
  struct DLL_node* prev;
};
typedef struct DLL_node DLL_node_t;

typedef struct {
  DLL_node_t* head; // pointer to the first node
  DLL_node_t* tail; // pointer to the last node
} DLL_t;
```

Implement the following functions for *Doubly Linked List* of ints.
The running time in parts a-c of must be O(1).

a) [6 points] Write a function that adds a new node with a given value to the head of the list.

```
void add_to_head() {DLL_t* list, int value) {









}
```

b) [6 points] Write a function that removes the first node of the list, and returns its value.

```
int remove_from_head() {DLL_t* list) {









}
```

c) [6 points] Write a function that removes a given node from the list.

```
// assumption node indeed belongs to the list
void remove_node() {DLL_t* list, DLL_node_t* node) {

}
```

d) [7 points] Write a function that gets a Doubly Linked List and a predicate p. The function removes all nodes for which p(node->value)==false.
For example, if we apply it on the list *head→[1←→2←→5←→4←→8]←tail* with p=is_even(), then the remaining list should be *head→[2←→4←→8]←tail*.
*Remember to release the memory of the deleted nodes.

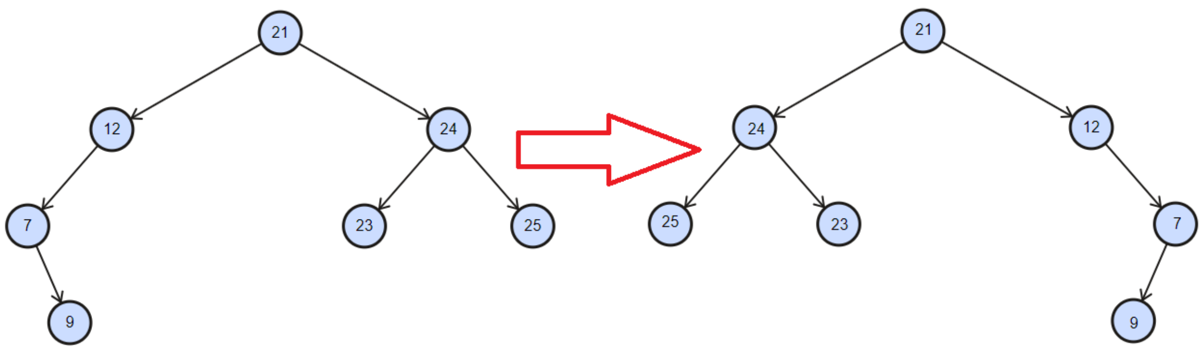
```
void filter() {DLL_t* list, bool(*p)(int)) {

}
```

Problem 3 [25 points]

In this problem use the following struct representing a node in a Binary Tree of ints.

```
struct BTnode {
  int value;
  struct BTnode* left;
  struct BTnode* right;
  struct BTnode* parent;
};
typedef struct BTnode BTnode_t;
```

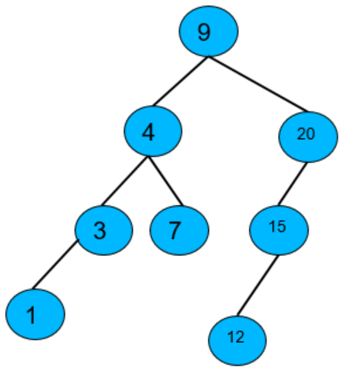
a) [7 points] Write an algorithm that gets a **Binary Tree** and converts it into its mirror reverse. For example



```
void mirror_tree(BTnode_t* root) {











}
```

[3 points] Explain the running time of your function.

b) [12 points] Write a function in C that gets a pointer to a node in a **Binary Search Tree**, and finds its predecessor. If the node is the minimal element in the tree, the function will return NULL. In the tree below: the predecessor of 4 is 3, the predecessor of 12 is 9, the predecessor of 7 is 4.



```
BTnode_t* find_predecessor(BTnode_t* node) {









}
```

[3 points] Explain the running time of your function.

Problem 4 [25 points]

a) [12 points] Write a function that gets an array of ints of length n, and outputs the length of the longest contiguous increasing subsequence in O(n) time. For example, on input [6,2,5,3,6,8,9,1] the answer should be 4, as the longest increasing subsequence is [3,6,8,9].
* If your solution runs in quadratic time or slower, you will get 8 points.

```
int longest_incr_subsequence(const int* arr, int n) {

}
```

[3 points] Explain the running time of your function.

b) [10 points] Write a function that gets an array of digits (given as ints) of length n, and returns a string containing the largest number possible using these digits. For example:

- on input [6, 7, 5, 3, 8, 3, 0, 0] the function should return “87653300”.
- on input [1, 2, 3, 4, 4] the function should return “44321”.
- on input [0, 0, 0, 0, 0, 0] the function should return “0”

For full marks solve the problem in O(n) time.

* You may assume all numbers in the array are between 0 and 9.

* Note: you need to return the number in a string because it may be too large to fit in an int/long.

** Remember to use dynamic memory allocation properly.

```
char* print_max_number(const int* arr, int n) {
}
```

Extra page

Empty page