EE 453/717: Advanced Computing for Electrical Engineers

Indian Institute of Technology Bombay

Autumn 2010

120 minutes

Midsem: **25 points + 3 bonus points** September 15, 2010

---

1. **[4 points]** Write a C++ function to merge two linked lists of sorted integers into a single linked list of sorted integers. Given pointers a and b to head nodes of two linked lists which each contain integers in decreasing order, the function needs to return a pointer to the first node of a linked list which contains the integers from both lists in decreasing order. For example,

   ```
   a : 6 5 3
   b : 4 3 1
   Merged list : 6 5 4 3 3 1
   ```

   So the function declaration is the following

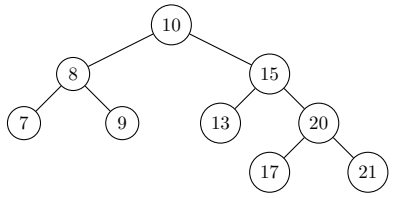
   ```
   node* MergeLists(node *a, node *b);
   ```

   Here `node` is a structure given by

   ```
   struct node
   {
       int x;
       node *next;
   }
   ```

2. (a) **[2 points]** Write a C++ function which takes a pointer to a node in a binary search tree (BST) and performs a **right rotation**, i.e. it exchanges the node and its left child while preserving the BST structure.

   (b) **[2 points]** Insert the integer 18 into the following BST such that it appears at the root of the modified tree.



   (c) **[3 points]** Consider a representation of a BST consisting of three arrays:

   i. The first array contains the nodes present in the BST in some arbitrary oder. The index of the root of the BST in this array is specified.

   ii. The second array is an integer array where the ith location contains the index of the left child of the node at location i of the first array.

   iii. The third array is an integer array where the ith location contains the index of the right child of the node at location i of the first array.

Index	0	1	2	3	4	5	6	7	8
Node array	10	8	15	7	9	13	20	17	21
Left child array	1	3	5	3	4	5	7	7	8
Right child array	2	4	6	3	4	5	8	7	8

For example, the BST of part (b) can be represented by the above arrays with the index of the root equal to zero in the node array. Given this representation of a BST, write pseudocode for the **right rotation** operation.

3. **[5 points]** We want to construct a data structure which will store all the unique CPIs of a batch of students. A CPI is defined to be a non-negative number less than or equal to 10.0 which has two digits after the decimal point. The data structure needs to support the following operations

   (a) Insert a given CPI into the list of unique CPIs.

   (b) Remove a given CPI from the list of unique CPIs.

   (c) Check if a given CPI is in the list of unique CPIs. Return `true` if it exists in the list and `false` otherwise.

   Implement the above operations in C++ with the restriction that the amount of storage used to store the list must be **at most 150 bytes**. Note that a short integer occupies 2 bytes and the boolean type `bool` occupies a byte.

4. **[4 points]** Write a C++ function to return a pointer to the nth node from the end of a linked list. Do it without calculating the length of a linked list. So the function declaration is the following where `head` points to the first node of the linked list.

```
node* ReturnNthFromEnd(node *head, int n);
```

5. **[5 points]** Write a C++ function to check if a given binary tree is a BST or not. The function declaration is the following.

```
bool IsBinaryTree(node *head);
```

6. **[3 bonus points]** Answer one of the following two questions for 3 bonus points.

   (a) Given an array of N integers, finding the maximum or minimum value requires $N-1$ comparisons each. Suppose we want to simultaneously calculate the maximum and the minimum valued integers in the array. Propose an algorithm which requires less than $2(N-1)$ comparisons and calculate the number of comparisons it requires. *Hint:* Consider pairs of integers from the array.

   (b) Explain the functionality of the following program.

```
#include <iostream>
#include <stack>

using namespace std;

#define EXIT_ADDR 11
```

```
#define ENTRY_ADDR 22
#define ADDR_A 33

int mysteryfunction(int n) {
  int result = 0;
  int m;
  int ret_addr;
  stack<int> st;
  st.push(EXIT_ADDR);
  st.push(n);
  ret_addr = ENTRY_ADDR;

  while ( true ) {
    switch (ret_addr) {
      case ENTRY_ADDR :
        m = st.top( );
        st.pop( );
        if (m == 1) {
          result = 1;
          ret_addr = st.top( );
          st.pop( );
        } else {
          st.push(m);
          st.push(ADDR_A);
          st.push(m-1);
        }
        break;
      case EXIT_ADDR :
        cout << "result is " << result << endl;
        return result;
        break;
      case ADDR_A :
        m = st.top( );
        st.pop( );
        result = m * result;
        cout << "result is " << result << endl;
        ret_addr = st.top( );
        st.pop( );
        cout << "return_address is " << ret_addr << endl;
        break;
    }
  }
}

int main() {
  int result;
  result = mysteryfunction(8);
  cout << result << endl;
}
```