

SECTION B (2 QUESTIONS, 38 MARKS)

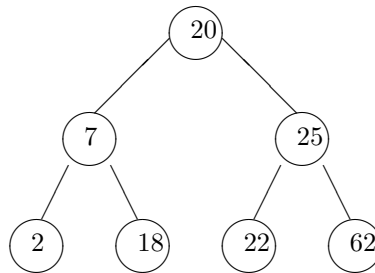
Use a separate book for Section B

For this section you may assume the following type definitions for a binary tree.

```
struct node
{
    int item;
    node *LChildPtr;
    node *RChildPtr;
};
typedef node *tree;
```

3. (18 marks in total.)

Consider the following diagram of a Binary Tree.



Now consider the following function defined for binary trees.

```
int mystery(tree tp) {
    if (tp == NULL) return 0;
    else {
        int l= mystery(tp->LChildPtr);
        int r= mystery(tp->RChildPtr);
        if ((tp-> item)%2 == 0) return max(max(l,r),tp->item);
        else return min(min(l,r),tp->item);
    }
}
```

$\max(x,y)$ returns the maximum of x and y , and $\min(x,y)$ returns the minimum of x and y .

- (4 marks) What value does `mystery` return when passed the above binary tree?
- (3 marks) How many recursive calls are required to compute the result in part (a)? What is the worst-case complexity in big Oh-notation relative to the number of nodes? (Count the number of recursive calls.)

- (c) (3 marks) State the definition of a binary search tree.
- (d) (5 marks) Now assume that `mystery` is only used for *binary search trees*. Change the implementation of `mystery` so that it produces exactly the same output, but more efficiently for some binary search trees.
- (e) (3 marks) For which binary search trees does your implementation in part (d) perform most efficiently? What is the worst-case complexity for those binary search trees?

4. (20 marks in total.)

Consider the following lists of items of a given **binary search tree** `tr` which were produced respectively using in-order traversal, pre-order traversal and post-order traversal. (See below for their implementations.)

- (i) 2, 3, 4, 5, 6, 8, 9 (output of `inOrder(tr)`)
- (ii) 5, 3, 2, 4, 6, 9, 8 (output of `preOrder(tr)`)
- (iii) 2, 4, 3, 8, 9, 6, 5 (output of `postOrder(tr)`)

Now answer the following questions.

- (a) (3 marks) Which item is contained in the root of `tr`? Give reasons.
- (b) (3 marks) Which items are contained in the left subtree of `tr`? Give reasons.
- (c) (3 marks) Which items are below 9 in `tr`? Give reasons.
- (d) (3 marks) Using your answers above (or otherwise), sketch a diagram of `tr`.
- (e) (5 marks) Now implement an iterative program `buildTree` that takes an array of integers and builds a binary search tree `bt` whose items consist of the integers in the array `A`. You may assume a function `insert` (see below for the specification) which inserts a node into a binary search tree. **Do not write an implementation for `insert`.**

```
void buildTree(int A[], int N, BST & bt) ;
// PRE:  bt is an empty tree;
//       A consists of distinct values; the length of A is N.
// POST: bt is a binary search tree with node values corresponding to A.

void insert(int n, BST& btt);
// PRE:  btt is a binary search tree;
// POST: inserts n into a leaf of btt, preserving the binary-search tree property,
//       but only if n is not already an item.
```

- (f) (3 marks) What order of integers would you use for the array `A` so that `buildTree(A, bt)` would produce a copy of the tree `tr`? Give reasons. (Choose from one of (i), (ii) or (iii).)

```
void preOrder(tree tp) {
    if (tp != NULL) {
        cout << tp->item << ",";
        preOrder(tp->LChildPtr);
        preOrder(tp->RChildPtr);
    }
}
```

```
void inOrder(tree tp) {
    if (tp != NULL) {
        inOrder(tp->LChildPtr);
        cout << tp->item << ",";
        inOrder(tp->RChildPtr);
    }
}
```

```
void postOrder(tree tp) {
    if (tp != NULL) {
        postOrder(tp->LChildPtr);
        postOrder(tp->RChildPtr);
        cout << tp->item << ",";
    }
}
```

End of Section B
Use a SEPARATE BOOK for Section C