Review problems for Exam 2:

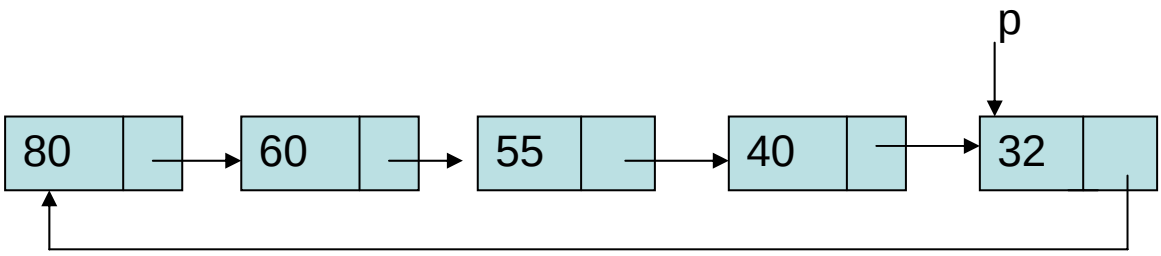
1. The Tower of Honoi problem can be solved using the following function where n denotes the number of disks on the first tower.

```
void movetower ( n, start, finish, temp)
{
        if ( n==1) {
                movesingle ( start, finish);
        }  else  {
                movetower ( n-1, start, temp, finish);
                movesingle ( start, finish);
                movetower ( n-1, temp,  finish, start);
}
```

Write the recurrence relation for this function and solve it to work out the time complexity.

2. Write a recursive function to find the digital root of an integer. The **digital root** of a number is the number obtained by adding all the digits, then adding the digits of that number, and then continuing until a single-digit number is reached. Thus the digital root of 5297 is 5, computed as follows:
5 + 2 + 9+ 7 = 23
2 + 3 = 5
[Hint: digital root of 5297 = digital root (7 + digital root of 529)]

3. Write a function to add 8 to all the node values of a circular linked list, and print the contents of each node.



4. What list is printed by the function when used on the following list:

```
 10 22 22 30 45 45 45 56 60 75 88 92 92
```

```
void function4(struct node *pList)
{
     struct node *afternext,  *pCur;
          pCur = pList;
     while(pCur->next !=NULL)
     {
        if(pCur->data == pCur->next->data)
          {
               afternext = pCur->next->next;
               pCur->next = afternext;
               free(afternext);
           }
        else
                pCur = pCur->next;
     }

     PrintList (pList);
}
```

5. Write a recursive function to form a linked list by adding items at the end of the list:

6. Each node of a binary tree with root p stores an integer value. The following function is supposed to examine the binary tree, and increase by 10 the value stored in nodes having two children and return the tree. Complete the code.

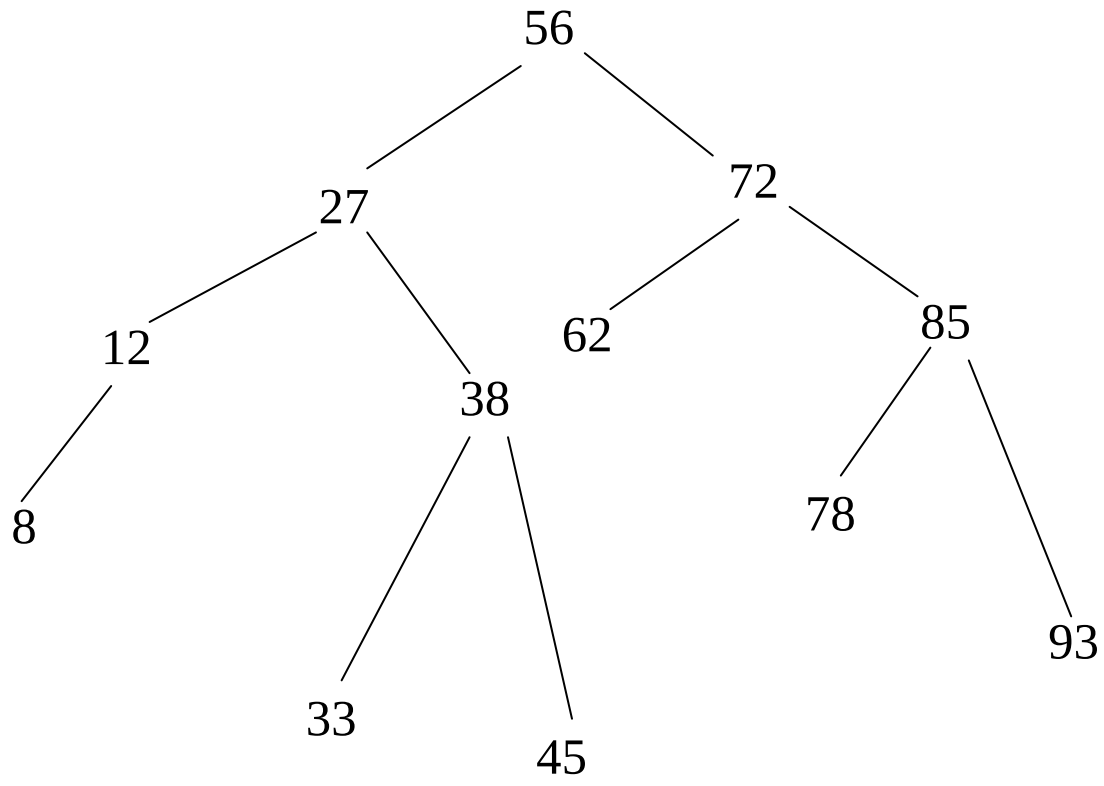
```
struct treenode* function1(struct treenode * p)
{
     if (p              )
     {
          if (p->left !=NULL && p->right !=NULL)



          function1(p->right);
     }
     return p;
}
```

7. Write a function which returns the smallest element in a Binary Search Tree

8. Write the inorder and preorder traversals of this tree:



9. Write a function to print the contents of a binary search tree in decreasing order.

10. Write a function which returns the copy of a given binary tree.