More Group HW

The following code is contained in the file ex1stck.h. Fill in the blanks with the C++ statement(s) that will correctly finish the method. Each blank may be filled in with more than one statement.

```
#ifndef Stackh
#define Stackh

#include <cstdlib>
using namespace std;

class Stack {
public:
      Stack (int MaxStackSize = 10);
      ~Stack() {delete [] stack;}
      bool IsEmpty() const {return top == -1;}
      bool IsFull() const {return top == MaxTop;}
      int Top() const;
      void push(int x);
      void pop();
private:
      int top;    // current top of stack
      int MaxTop; // max value for top
      int *stack;
};

Stack::Stack (int MaxStackSize)
{
        MaxTop = MaxStackSize - 1;
        stack = new int[MaxStackSize];
        top = -1;
}

int  Stack::Top( ) const
{
     if (______________)
     {
          cerr << "Error";
          exit(1);
     }
      return ____________________;
}

void Stack::push(int x)
{
        if (_________) {
                cerr << "error";
                exit(1);
        }
```

```
          ____________________
}

void Stack::pop()
{
          if (_________________) {
                  cerr << "Error";
                  exit(1);
           };
           ___________________
}

#endif
```

b) (5 points) What is printed by the following program segment given the stack header file ex1stck.h above.

```
Stack s(5);
int x;

s.push(12);
s.push(6);
s.push(10);

while (!s.IsEmpty())
{
      x = s.Top();
      cout << x << ' ';
      s.pop();
}
```

c) (5 points) Use the above stack class definition to write the code that would set an integer `x` to the second element from the top of the stack. Print an appropriate error statement if this is not possible (i.e. the stack has less than two elements).

9. (12 points) Fill in the blank

a) LIFO stands for ________________________.

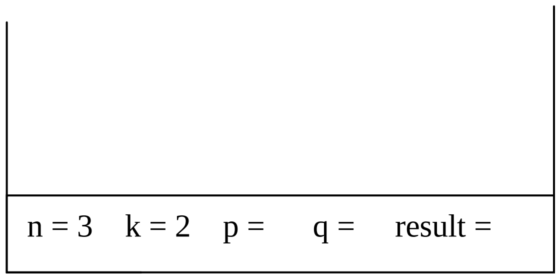
b) In a linked stack, the top node points to ___________________________.

What is printed by the following C++ program?

```
#include <iostream>
#include <stdexcept>
#include "stack2.h"  //array implementation of a stack
using namespace std;

int
main ()
{
      Stack<int> A(5);
      int i,x;

      try{
            for (i = 0; i < 6 ;i++)
                  A.push(i + 5);

            for (i = 0; i < 6; i++)
            {
                  x = A.Top();
                  A.pop();
                  cout << x << ' ';
            }

      }

      catch (logic_error le){
            cout << le.what() << endl;}
      catch (...){cout << "An exception has occurred" << endl;}

      return(0);

}

/*Given the following header and methods for push and pop

template <class StackItemType>
class Stack
{
public:
   Stack(int maxstack);
   bool isEmpty() const;
   void push(const StackItemType& newItem) throw(StackException);
   void pop() throw(StackException);
   void pop(StackItemType& stackTop) throw(StackException);
   void getTop(StackItemType& stackTop) const
      throw(StackException);

private:
   /** Array of stack items */
   StackItemType *items;
   int           top;
   int MAX_STACK;
};
```

```
template <class StackItemType>
void Stack<StackItemType>::push(const StackItemType& newItem)
{
// if stack has no more room for another item
   if (top >= MAX_STACK-1)
      throw logic_error("StackException: stack full on push");
   else
   {  ++top;
      items[top] = newItem;
   }  // end if
}  // end push

template <class StackItemType>
void Stack<StackItemType>::pop() throw(StackException)
{
   if (isEmpty())
      throw logic_error("StackException: stack empty on pop");
   else
      --top;     // stack is not empty; pop top
}  // end pop

*/
```

1. Given two numbers, n and k, the binomial coefficient for these two numbers are equal to the number of combinations of n choices given k slots. The following code calculates binomial coefficients recursively. Show how the operating system stack changes when the recursive function *binoc* executes. The first call to *binoc* is done for you.



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

int binoc (int, int);

int
main()
{
      int n, k, result = 0;
      n = 3; k = 2;

      result = binoc (n,k);

      cout << "result = " << result << endl;
      return 0;
}
```

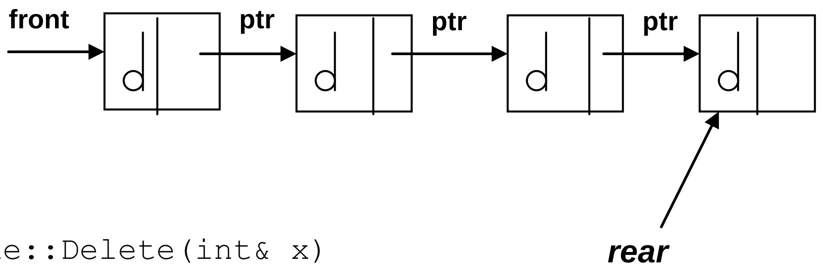
```
int binoc (int n, int k)
{
      int p, q, result;

      if (k == 0) return 1;
      if (k == n) return 1;

      p = binoc(n - 1, k - 1);
      q = binoc (n - 1, k);
      result = p + q;
      return result;
}
```

2. The power function can be written recursively as follows:

```
      pow (n, 1) returns n
      pow (n, 0) returns 1

      pow (n, k) = n * pow (n, k-1)
```

Write the code that implements recursive function pow.

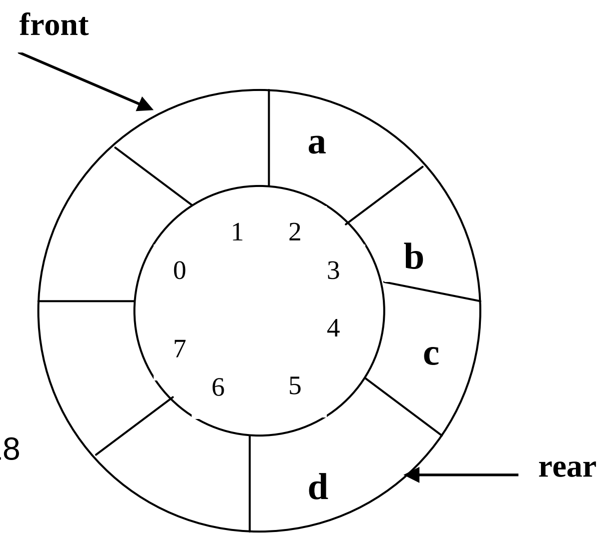
You are given the following **queue**, where each node class contains an integer variable called *d* and a pointer variable called *ptr*. Complete the code for the delete function which deletes the first node of a queue and places the value from the data portion of node into x.



```
void Lqueue::Delete(int& x)
{
      if (_______________)
      {
            cout << "error";
            exit(1);
      }
   x = _______________;

   Node *p = ____________;

   front = _______________;

   delete _____;
}
```

Assume full or empty situation will print some type of error message and will not exit the program. The function enque, adds an element to the queue, dequeue deletes an element form the queue, and print will print the entire queue starting at the front. Draw pictures to illustrate each step.

O.enqueue (z);
Q.enqueue (f);
Q.enqueue (g);
Q.dequeue (x);
Q.dequeue (x);
Q.enqueue (m);
Q.print();



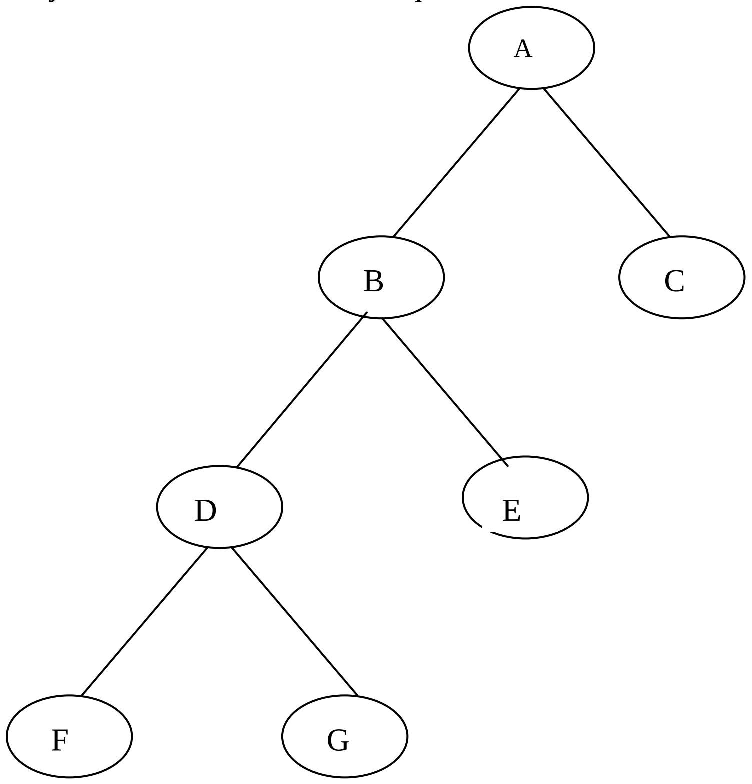
More to do....
Carrano pg 495 - 496 8,9,10,16,18

See next two pages for optional questions

Do in Malik, Edition 6 -
7 a, b, c 8 a, b, c 9 b, c 15 16

Optional GHW tree questions of the type that will be on the exam (we will go over these in class)

) Base your answers to each of the questions below on the following binary tree.



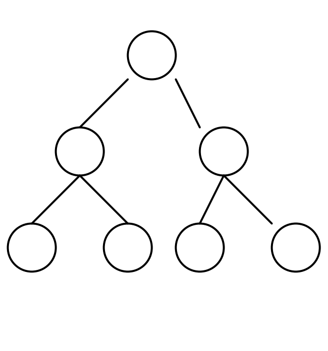
a) List all the **leaves** in the tree.

b) List the nodes which are the **children** of node B

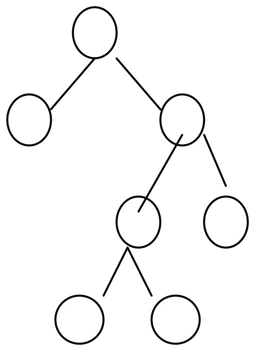
c) List an **ancestor** of node C.

d) What is the **height** or depth of the tree?

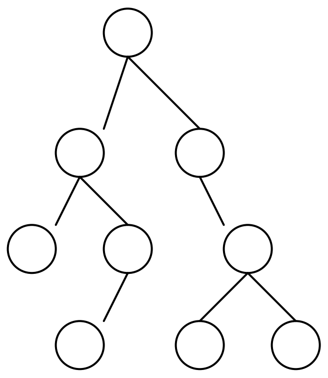
Consider the following binary trees:



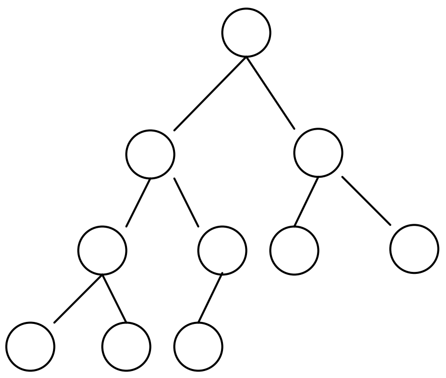
Tree A



Tree B



Tree C



Tree D

a) Which tree(s) are strictly binary?
b) Which tree(s) are complete?
c) Which tree(s) are full?