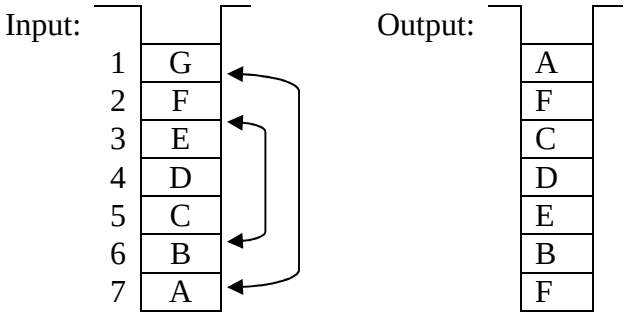
Examples:

Q1) Write a function which reverses the odd items in a stack counted from top e.g .



Solution :

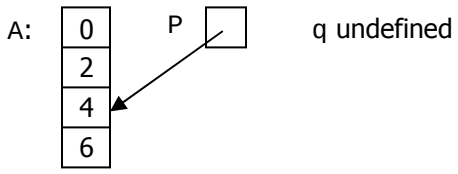
```
template <class T>
void Reverse_Odd_Items(Stack <T> &X)
{  Stack<T> Odd_S, Even_S, Temp_Odd_S;
   int odd  /* set odd=1 if number of items in stack is odd, odd=0 otherwise  */

   /* return without doing anything if input stack is empty
   if (!X.stackEmpty( ))
  {   /*at least one element */
      /* fist seperate odd and even itms into their respective stacks */
      while (!X.StackEmpty( ))
      {    Odd_S.Push(X.Pop( ));
           odd=1;
           if (!X.StackEmpty( ))
           {     /* if still there is an even element after odd one popped*/
                Even_S.Push(X.Pop( ));
                 odd=0;
            }
      }
     /* reverse odd elements a second time */
     while (!Odd_S.StackEmpty( ))
           Temp_Odd_S.Push(Odd_S.Pop( ));
      /* Now re-insert into  original stack, odd ones in reverse order,
          even ones in original order */
       if (odd= =1) X.Push(Temp_Odd_S.Pop()); /* bottom one is odd */
       while (!Temp_Odd_S.StackEmpty( ))
       {   /* odd is empty iff  even is empty */
           X.Push(Even_S.Pop( ));
           X.Push(Temp_Odd_S.Pop( ));
        }
   } /* end if original stack non_empty */
} /* end function */
```

Q2) Consider the following C++ program section and Show the contents of p, q and the array A at:

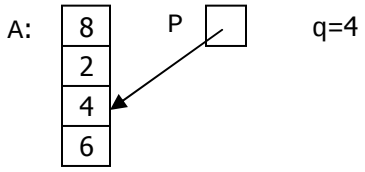
(a) Control point 1
(b) Control point 2
(c) Control point 3

```
int S(int a, int &b)
{       int t;
        t=a;
        a=b*2;
        b=t*2;
        return t; }
void main(void)
{       int *p;
        int q, A[4];
        for (int i=0; i<4; i++)
                A[i]=i*2;
        p=A+2;
        // control point 1
        q=S(*p,A[0]);
        // control point 2
        S(A[0],*p);
        // control point 3
}
```

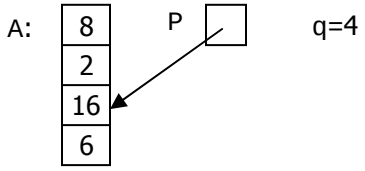
a) At Control point 1:



(b) At Control point 2:
S(4 by value, 0)
t=4
local a=0*2=0
A[0]←8



(c) At Control point 3:
S(8 by value, 0)
t=8
local a=4*2=8
// note that the 2nd parameter of function
// S is passed by reference, thus
A[2]←8*2=16



Q3)
(a) What is the order (time complexity in O(.) notation) of the following tasks in the worst case?

i. computing the sum of the first n even integers by using a for loop
ii. pushing an item to a stack of n items

Solution:

(a) i. computing the sum of the first n even integers by using a for loop

```
for (i=1;i<=n;i++) sum=sum+i*2;
```

the work (time it takes…) grows linearly with n, hence O(n)

ii. pushing an item to a stack of n items

Recall the implementation of Push method of Stack class

```
void Stack::Push(const DataType& item)
{
    if (top==MaxStackSize-1)
    {
        cerr<<"Stack overflow"<<endl; exit(1);
    }
    top++;
    stacklist[top] =item;
}
```

work done in pushing an item to a stack is independent of stack size…Hence, O(1)

(b) Order the following functions by growth rate: *nlog n*, *nloglog n*, $log^2\ n$, *nlog* n^2, $2^{n/2}$, 2^n, *37*, $n^2 log\ n$, *2000*. Indicate the group of functions which grow at the same rate.

Solution

37, *2000*
$log^2\ n$
nloglog n
nlog n, *nlog* n^2
$n^2 log\ n$
$2^{n/2}$,
2^n

(c) Suppose that the implementation of a particular algorithm appears in C++ as follows:

```
for (int i=1; i>=n; i++)
{   for (int j=1; j>=i; j++)
    {   for (int k=1; k>=10; k++)
            myfunction(i,j,k,n);
    }
}
```

What is the time complexity of the algorithm, assuming that myfunction has a running time, which is O(log n)? Justify your answer.

Solution:

If myfunction requires t time units, the innermost loop on k requires 10*t time units. The loop of j requires 10*t*i time units and the outermost loop on i requires:

$$\sum_{i=1}^{n}(10*t*i) = 10*t*(1+2+\ldots+n) = 10*t*n*\frac{(n+1)}{2} = O(t*n^2)$$

Since t is O(log n) then the overall execution time is $O(n^2 \log n)$

Q4) Determine, showing your arguments briefly, the O( ) complexities of:

(a) The function $(n^3 + 7) / (n + 1)$

Solution: $(n^3 + 7) / (n + 1) = n^2 - n + 1 + 6/(n+1) \rightarrow O(n^2)$

(b) The running time of the following *C++* code, as a function of *n* :

```
for (h=1; h<n; h++)
    for (i=1; i<n; i++)
        for (j=1; j<i*i; j++)
            count++;
```

(Note that $1^2 + 2^2 + 3^2 + ... + n^2 = n (n + 1) (2n + 1) / 6$)

Solution:

The running time of the following *C++* code, as a function of *n* :

```
for (h=1; h<n; h++)
    for (i=1; i<n; i++)
        for (j=1; j<i*i; j++)   } exec-time  sum_{i=1}^{n} i^2
            count++;
```

execution time=$n*n (n + 1) (2n + 1) / 6) \rightarrow O(n^4)$

(c) The worst case running time of the following algorithm, described informally, as a function of n :

Search a sorted array of n elements for a given *key*:

1. Start with size ← full table size.
2. Partition the table to be searched into three.
3. If the table is too small for this (i.e., if size < 3) simply check its contents and exit either with success or with failure.
4. If *key* < the last element of the lowest partition, from now on, consider only this lowest partition and go back to step 2. (In steps 4 and 5, if the key is found at the comparison operation, exit the algorithm with success.)
5. If *key* < the last element of the middle partition, from now on, consider only this middle partition and go back to step 2.
6. Otherwise from now on, consider only the top partition and go back to step 2.

Solution:
n items can be partitioned into 3, at most $\lceil \log_3 n \rceil$ times.. → O(logn)

Q5) Given the following C++ program:

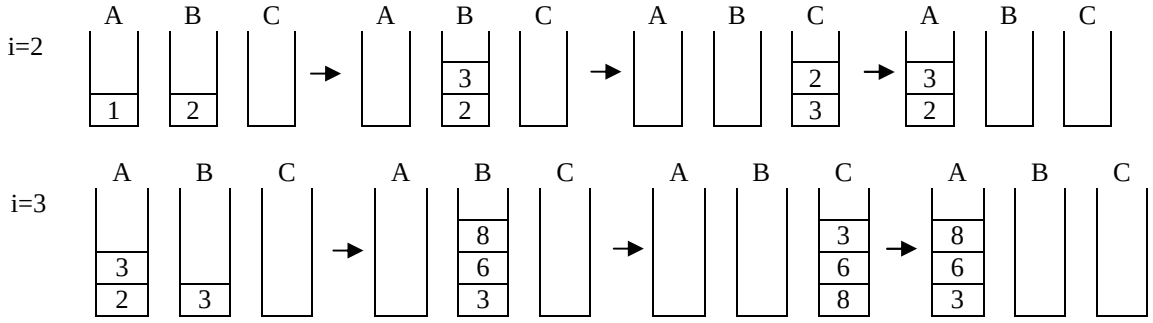
```
template <class T>
Class Stack
{
        private:
                T stacklist[MaxStackSize];
                int top;
        public:
                Stack(void);
                void Push(const T& item);
                T Pop(void);
                void Clearstack(void);
                T Peek(void) const;
                int StackEmpty(void) const;
                int StackFull(void) const;
};

void myfunction(int n)
{
  Stack<int> SA, SB, SC;
  int prev, current;
```

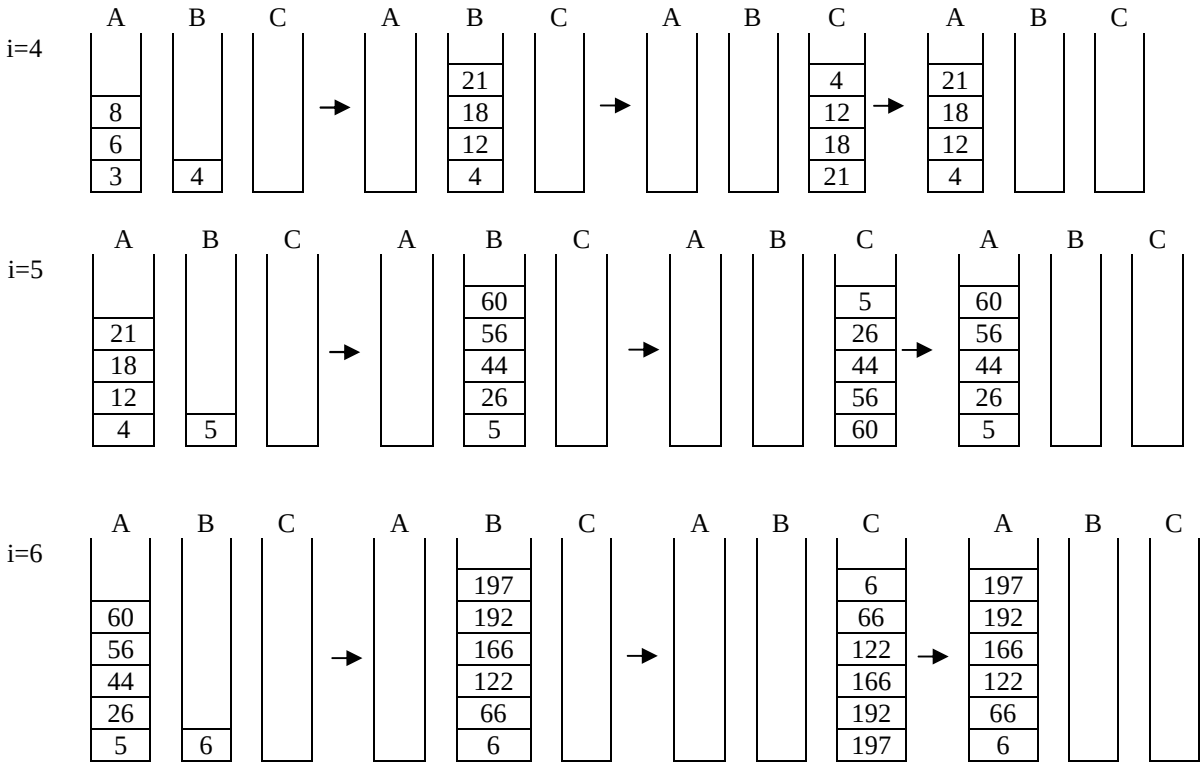
```
  SA.Push(1);
  for (int i=2; i<=n; i++)
  {
        cout<endl;
        cout<<“i=“<<i<<“::”;
        prev=i;
        SB.Push(prev);
        cout<<SB.Peek<<“,”;
        while !(SA.StackEmpty())
        {
          current=prev + SA.Pop;
          SB.Push(current);
          cout<<SB.Peek<<“,”;
          prev=current;
        }
        while !(SB.StackEmpty())
        {  SC.Push(SB.Pop); }
        while !(SC.StackEmpty())
        {  SA.Push(SC.Pop); }
   }
}
```

(a) What will be the output if myfunction is called with n=6, i.e., myfunction(6) ?

Solution:
myfunction(6)

i=2::2,3,
i=3::3,6,8,
i=4::4,12,18,21,
i=5::5,26,44,56,60
i=6::6,66,122,166,192,197





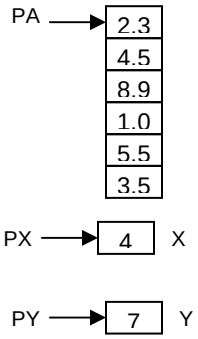
(b) What is the complexity of the algorithm in O(.) notation in terms of n?

Solution: O(n*n)

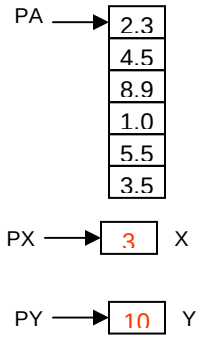
Q6) Give the final values of X, Y and A after the following C++ statements are executed:

```
void main()
{
        int X=4, Y=7, *PX=&X, *PY;
        float A[]={2.3, 4.5, 8.9, 1.0, 5.5, 3.5}, *PA=&A[0];
        PY=&Y;   /* check pt 1 */
        (*PX)--;
        *PY+=*PX; /* check pt 2 */
        PY=PX;
        *PY=55;
        *PA+=3.0; /* check pt 3 */
        PA++; /* check pt 4 */
        *PA++=6.8; /* check pt 5 */   <<<< executed as  *PA=6.8; PA++;
        PA+=2;  /* check pt 6 */
        *++PA=3.3; /* check pt 7 */   <<<< executed as PA++; *PA=3.3;
}
```

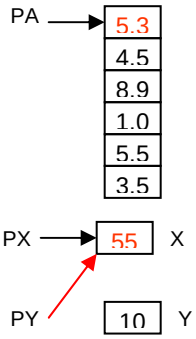
check pt 1



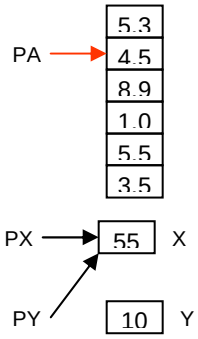
check pt 2



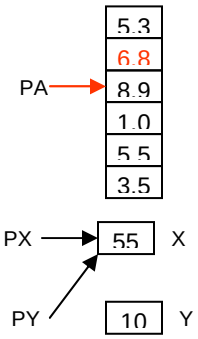
check pt 3



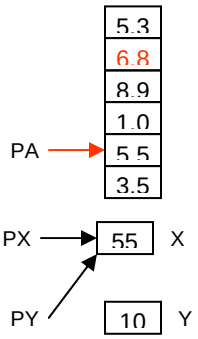
check pt 4



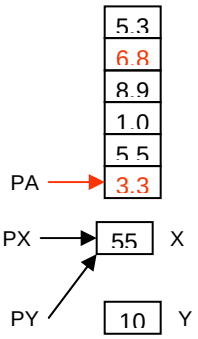
check pt 5



check pt 6



check pt 7



Q7) Given the partial class definition

```
class MyClass
{  private:
     float a[50];
  public:
     MyClass(void);
     ??? Fun(??? J); // single input argument};
```

Implement the member function Fun which will allow the user to insert floating numbers to the desired location in the floating array a. Array indices greater than 49 will be inserted in the last location and negative indices will be inserted in the 1st location in the array. Any index between 0-49 will be inserted in the corresponding location in the array. This member function will take only a SINGLE input argument. Comment on each line of your code

Solution:

In this question, your function should be able to put a floating number x in the array location y

```
float& MyClass::Fun(int j)
{
        int index;
        index=j;
        if (j<0)
                index=0;
        if (j>49)
                index=49;
        return a[index];
}
```

An example use of this function in a main program is

```
MyClass A;
int x=10;
float y= 1.234;
A.Fun(x) = y;
```

Q8) Given the main program and function Fun

```
void Fun(int *a, int *b)
{   a=a+5;
   b=a-2;};
main ()
{
   int a[50], b[50];
   int *p, *q;
   p=a+3;
   q=b+5;
   Fun(p,q);  // Line (X)}
```

a) What are the contents of p and q after the execution of line (X)?
Solution: Since the parameters are passed by value, the contents of p and q are unchanged.

b) Propose a way how Fun should be modified in order to make p point to the first element in a and q point to the first element in b. Also indicate how Fun should be called in line(X)

```
void Fun(int **a, int **b)
{   *a=*a-3;
   *b=*b-5;};
main ()
{
   int a[50], b[50];
   int *p, *q;
   p=a+3;
   q=b+5;
   Fun(&p,&q);  // Line (X)}
```

Q9) (a) For the following declaration of the Stack class, assuming that the public methods Stack, Push, Pop, StackEmpty and StackFull are implemented as discussed in class, implement the C++ member function Count that will return the number of items in the owner object. Your member function should access the data structure through the existing member functions.

```
template <class T>
class Stack
{private: T stacklist[Maxstacksize];
        int top;
public: Stack(void);
         void Push(const T &item);
         T Pop(void);
         int StackEmpty(void) const;
         int StackFull(void) const;
         int Count(void) const;
```

Solution:
(with the ability of member function to access private data members)

```
         template <class T>
         int Stack<T>::Count(void) const
         {        return (top+1);}
```

(b) Using the Stack class definition in part (a), implement a global function

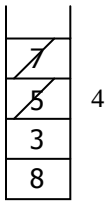
```
                  template <class T>
                  int Bottom (Stack<T> &s, T &last)
                  { }
```

that returns 0 if stack s is empty and 1 if it is non-empty, assigning the bottom element to the argument last. Upon return from your function the original stack should remain unchanged. Your function implementation should be properly commented for understandability.
Solution:

```
template <class T>
         int Bottom(Stack<T> &s, T &last)
         {        Stack<T> temp_s; // temp_s will store the original stack in reverse order
                  // first reverse original stack into temp_s
                  if (s.StackEmpty) { return 0;}
                  else {
                    while(!s.StackEmpty())
                    { temp_s.Push(s.Pop());};
                    // now, bottom of original is top of temp_s
                    last=temp_s.Pop();    // this is the wanted item
                    s.Push(last);  // restore bottom
                    while (!temp_s.StackEmpty())        // restore the rest
                    { s.Push(temp_s.Pop());};
                    return 1;
                  }
         }
```

(c) Still using the Stack class definition in part (a), what is the output from the following sequence of operations?

```
Stack<int> S;
int x=3, y=5, z=7;
s.Push(8);
s.Push(x);
s.Push(y);
s.Push(z);
x=s.Pop();
s.Pop();
s.Push(4);
cout<< x << endl;                    // output1: ……7…….
cout<< s.Count()<<endl;              // output2: ……3…….
y=s.Pop();
z=s.Pop();
cout<< y << endl;                    // output3: ……4…….
While (!s.StackEmpty())
        x=s.Pop();
cout<<x<<endl;                       // output4: ……8…….
```



Q10) You are given the Stack and Queue classes covered in lectures:

```
template <class T>
class Stack
{
private:
  T stacklist[MaxStackSize];
  int top;
public:
  Stack(void); void Push(const T& item);
  T Pop(void);
  void ClearStack(void);
  int StackEmpty(void) const;
  int StackFull(void) const;
};

template <class T>
class Queue
{
private:
  int front, rear, count;
  T qlist[MaxQSize] ;
public:
  Queue(void);
  void Qinsert(const T& item);
  T QDelete(void);
  int QLength(void) const;
  int QEmpty(void) const;
};
```

Write a function void findinQ (int key, Queue <int>& MyQ)
which searches a given key in a Queue and deletes it if it exists. If the key does not exist the function ends after the search without performing an operation on the queue. You can only use stacks and at most one temporary variable in this function.

Solution:

```
void findinQ (int key, Queue <int>& MyQ)
{
        Stack<int> S1;
        Stack<int> S2;
        int temp;
        while(!MyQ.QEmpty())
        {
                temp=MyQ.QDelete();
                if(temp!=key)
```

```
            S1.Push(temp);
        }
        while(!S1.StackEmpty())
            S2.Push(S1.Pop());
        while(!S2.StackEmpty())
            MyQ.QInsert(S2.Pop());
}
```

Q11) Given afollowing C++ programs; write down the output of each, indicate the erroneous program(s) if any. Justify your answers.

#include
<iostream.h>
 	
void triple(double &num);
 	
main()
{
  double d=10.0;
  triple(d);
  cout<<d;
  return 0;
}
 	
//Triple num's value
void triple(double &num)
{
** num=3*num;**
}
	#include
	<iostream.h>
	
	void triple(double &num);
	
	main()
	{
	  double d=10.0;
	  triple(&d);
	  cout<<d;
	  return 0;
	}
	
	//Triple num's value
	void triple(double &num)
	{
	  num=3*num;
	}
		#include
		<iostream.h>
		
		void triple(double *num);
		
		main()
		{
		  double d=10.0;
		  triple(&d);
		  cout<<d;
		  return 0;
		}
		
		//Triple num's value
		void triple(double *num)
		{
		  ***num=3**num;**
		}
			#include
			<iostream.h>
			
			void triple(double num);
			
			main()
			{
			  double d=10.0;
			  triple(d);
			  cout<<d;
			  return 0;
			}
			
			//Triple num's value
			void triple(double num)
			{
			  num=3*num;
			}

Solution: 30 (parameter passed by reference)	Solution: incorrect! (error: could not convert `&d' to `double&'	Solution: 30 (passed by reference with pointer implementation)	Solution: 10 (passed by value)