

ARRAYS, POINTERS, PARAMETER PASSING

Question 1 (25 points)

```
#include <iostream.h>
class SA
{ public:
    int *s, a;
    SA(int &o, int aa) {s=&o; a=aa;}
    void A(int &n) {*s=*s**s; n=a*n ;sa();}
    void B(int *n) {*s=*s**s; *n=a**n ;sa();}
    void C(int n) {*s=*s**s; n=a*n ;sa();}
    void D(int *n) { s=&a ; *n=a**n ;sa();}
    void sa() {cout <<"SA:"<<*s<<":"<<a<<endl;};
};
int main(void)
{
    int i=1, j=2, k=3;
    SA x(i,j);
    x.sa();
    x.A(k);
    x.A(x.a);
    x.B(x.s);
    cout<<k<<endl;
    x.C(k);
    x.D(&x.a);
    x.D(x.s);
}
// Write down the output of the program.
```

Given above C++ program; write down the output, indicate the erroneous step(s) if any. Justify your answer.

1: SA:1:2
2: SA:1:2
3: SA:1:4
4: SA:4:4
5: 6
6: SA:16:4
7: SA:16:16
8: SA:256:256
9:
10:

Question 2

Find out the output produced by the following programs

<pre>a) #include <iostream> void main () { int i; int numbers[6]; int * p; p = numbers; for (i=0; i<6;i++){ *(p+i)=i; } p = &numbers[2]; *p = 29; p = numbers + 3; *p = 17; p= numbers; *p = 21; p++; *p = 32; p = numbers; *(p+4) = 16; for (int n=0; n<6; n++) cout <<"Line"<<n<<": "<< numbers[n] <<"\n"; }</pre>	<p>Output:</p> <p>Line 0: 21 Line 1: 32 Line 2: 29 Line 3: 17 Line 4: 16 Line 5: 5</p>
<pre>b) #include <iostream> void main() { const int size=5; int B[size][size]; for (int r=0; r<size; r++) { for (int c=0; c<size; c++) { B[r][c]=10*r+c; } } for (r=1; r<size; r++) { for (int c=1; c<=r; c++) { if (r==c) { B[r][c]=1; } else { B[r][c]=0; } } } cout<< "Array B is"<<endl; for (r=0; r<size; r++) { for (int c=0; c<size; c++) { cout<<B[r][c]<<" "; } cout << endl; } cout << endl; }</pre>	<p>Output:</p> <p>Array B is 0 1 2 3 4 10 1 12 13 14 20 0 1 23 24 30 0 0 1 34 40 0 0 0 1</p>

Question 3 (25 points)

```
#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;
}
```

(A)

```
#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;
}
```

(B)

```
#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;
}
```

(C)

```
#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;
}
```

(D)

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

Program (A): Output: 30 (parameter is passed by reference)
Program (B): **incorrect !!** (error: could not convert '&d' to 'double&')
Program (C): Output: 30 (passed by reference with pointer implementation)
Program (D): Output: 10 (parameter is passed by value)

(b) Given the following C++ program:

```
#include <iostream.h>
void ArrayExchange(int ....a, int ....b)
{
    int k, temp;
    for (k=0;k<4;k++)
```

```

    {
        temp= ....b[k];
        ....b[k]= ....a[k];
        ....a[k]=temp;
    }
}
int main()
{
    int *pi, *pj, l;
    int i[]={0,1,2,3};
    int j[]={5,6,7,8};
    .... = ....;
    .... = ....;
    cout <<"Original\ti: ";for(l=0;l<4;l++) cout<<i[l]<<'\\t';cout<<endl;
    cout <<"Original\tj: ";for(l=0;l<4;l++) cout<<j[l]<<'\\t';cout<<endl;
    ArrayExchange(....pi,....pj);
    cout <<"Exchanged:\ti: ";for(l=0;l<4;l++) cout<<i[l]<<'\\t';cout<<endl;
    cout <<"Exchanged:\tj: ";for(l=0;l<4;l++) cout<<j[l]<<'\\t';cout<<endl;
}

```

Fill in the blanks, i.e., (. . .), with appropriate operators if required in order to make the program work as desired. Briefly justify your reasoning.

Solution:

```

#include <iostream.h>
void ArrayExchange(int *a, int *b)
{
    int k, temp;
    for (k=0;k<4;k++)
    {
        temp=b[k];
        b[k]=a[k];
        a[k]=temp;
    }
}
int main()
{
    int *pi, *pj, l;
    int i[]={0,1,2,3};
    int j[]={5,6,7,8};
    pi=i;
    pj=j;
    cout <<"Original:\ti: ";for(l=0;l<4;l++) cout<<i[l]<<'\\t';cout<<endl;
    cout <<"Original:\tj: ";for(l=0;l<4;l++) cout<<j[l]<<'\\t';cout<<endl;
    ArrayExchange(pi,pj);
    cout <<"Exchanged:\ti: ";for(l=0;l<4;l++) cout<<i[l]<<'\\t';cout<<endl;
    cout <<"Exchanged:\tj: ";for(l=0;l<4;l++) cout<<j[l]<<'\\t';cout<<endl;
}

```

The output of the program is:

Before:	i:	0	1	2	3
Before:	j:	5	6	7	8
After:	i:	5	6	7	8
After:	j:	0	1	2	3

(c) Briefly describe two main approaches to provide a function with argument along with their differences if any.

BY VALUE: The calling program copies the actual full object value to the local data area of the called program.

BY REFERENCE: The address of the object as stored by the calling program is passed into the called program which operates on the original data; not its local copy.

Question 4) a) (8 pts) Write the program outputs by tracing the following piece of code:

```
#include<stdio.h>
#include<iostream.h>
void main()
{
```

```
    int *a, c, d=5;
    int *b=&d;
    c=2;
    a=&c;
    *a=d+12;
    cout<<"a:"<<*a<<"b:"<<*b<<"\n";
    cout<<"c:"<<c<<"d:"<<d<<"\n";
    cout<<"a:"<<a<<"b:"<<b<<"\n";
    a=b;
    *b=*a+5;
    cout<<"a:"<<*a<<"b:"<<*b<<"\n";
    cout<<"c:"<<c<<"d:"<<d<<"\n";
    cout<<"a:"<<a<<"b:"<<b<<"\n";
```

Outputs:

```
*a:17*b:5
a:17d:5
a:address1b:address2
*a:10*b:10
c:17d:10
a:addressb:address
```

b) (17 pts) Write or draw diagrams to show the values of the variables (a,b,c,d,f,A) at the control point 1 and control point 2.

```
int F(int *x, int *&y,int&z)
```

```
{
    int *w=y;
    y=&z;
    *y=*(x+1);
    x[2]=(*w)*z;
    x=x+2;
    return *x;
}
```

```
void main(void)
```

```
{
    int a=20, b=100, f;
    int A[3];
    int *c=&b;
    int *d=A+1;
    for (int i=0; i<3; i++)
        A[i]=3*(i+1);
```

```
//control point 1
```

```
    cout<<"a:"<<a<<"b:"<<b<<"\n";
    cout<<"*c:"<<*c<<"*d:"<<*d<<"*A:"<<*A<<"\n";
    cout<<"A[0]:"<<A[0]<<"A[1]:"<<A[1]<<"A[2]:"<<A[2]<<"\n";
```

```
    f=F(A,c,a);
```

```
//control point 2
```

```
    cout<<"a:"<<a<<"b:"<<b<<"\n";
    cout<<"*c:"<<*c<<"*d:"<<*d<<"*A:"<<*A<<"\n";
    cout<<"A[0]:"<<A[0]<<"A[1]:"<<A[1]<<"A[2]:"<<A[2]<<"\n";
    cout<<"f:"<<f<<"\n";
```

```
}
```

Outputs:

```
a:20b:100
*c:100*d:6*A:3
A[0]:3A[1]:6A[2]:9
a:6b:100
*c:6*d:6*A:3
A[0]:3A[1]:6A[2]:600
f:600
```

Question 5. A) (6 pts) Below are some lines from C++ codes. State if the underlined C++ instructions are correct or not. Explain your reasoning.

a)

```
int a=50;  
char x[a];    //Correct/Not correct? Why? Not. a is not constant integer!
```

b)

```
void functionx(int *x)  
{  
    //do something  
}  
void main ()  
{ int A[]={0,1,2,3,4,5};  
    functionx(A);    //Correct/Not correct? Why? Correct. A is also a pointer to an integer.  
    //do something}
```

c)

```
char A[]={ 'a', 'b', 'c' };  
A = "Hi";    //Correct/Not correct? Why? Not Correct. An array cannot be assigned by a string.
```

Question 5. B) (7 pts) What will be the output of the following piece of code? When contents of a memory location are not known, indicate this.

```
int i, j, M[3][4];  
int temp;  
for(i = 2; i >= 0; i = i - 1)  
{ for(j = 3; j >= 0; j = j - 1)  
    { if (i==j) M[i][j]=10;  
      else      M[i][j]=i*j;  
      cout<< i<<" "<< j<<": "<< M[i][j]<<" "; }  
    cout<<endl; }  
temp=*(M+5);  
cout<<M[0][0]<<'and'<< temp << endl;
```

**2,3:6; 2,2:10; 2,1:2; 2,0:0;
1,3:3; 1,2:2; 1,1:10; 1,0:0;
0,3:0; 0,2:0; 0,1:0, 0,0:10;
10 and 2 //M[1][1] will be printed out, which is 2.**

Question 5. C) (6 pts) Class Rectangle is declared and implemented as below.

```
class Rectangle
{ private:
    float length;
    float width;
public:
    Rectangle(float l, float w)          /* constructor function */
    {   length=l;
        width=w; };
    float Circumference(void) const
    {   return 2*(length+width); };
    float Area(void) const
    {   return (length*width); };
}
```

What will be the output of the following piece of code? When contents of a memory location are not known, indicate this.

```
float a,b,c;
Rectangle *p;
Rectangle R[3]= { Rectangle (5.0, 7.0), Rectangle (5.0, 6.0), Rectangle(5.0, 6.0)};
p=R++;
a=*p.Area();
b=R[0].Circumference();
c=R[3].Area();
cout<<"a is "<<a<<" , "<<"b is "<<b<<" , "<<"c is "<<c<<".";
cout<<endl;
```

a is 30, b is 24, c is XX.

Question 5. D) (6 pts) Trace the following code and write the outputs.

```
main ()
{ int v[] = {9,8,7,6,5,4,3,2,1,0};
  for (int i=0; i<10; i++)
    cout << " " << v[i] << endl;
  bool changed;
  do{changed = false;
    for (int i=0; i<10; i++)
      {if ( v[i] > v[i+1] )
        { double temp = v[i];
          v[i] = v[i+1];
          v[i+1] = temp;
          changed = true;}}
    }while (changed);
  for (int i=0; i<10; i++) cout << " " << v[i]<< endl;
}
```

9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9

Question 6. (10 PTS) 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`

1st method: Use references

```
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;
```

2nd method: Use another class definition

```
class NewClass
{
    private:
        float number;
        int index;
```



```

    public:
        NewClass(float a, int b); // initializes number      with
                                a, index with b
        float GetNumber(void); // returns number
        int GetIndex(void); // returns index
}

void MyClass::Fun(NewClass j)
{
    int index;
    int temp;

    index = temp = j.GetIndex();
    if (temp < 0)
        index = 0;
    if (temp > 49)
        index = 49;

    a[index] = j.GetNumber();
}

```

Then an example usage in main is

```

MyClass A;
int x = 10;
float y = 1.234;
NewClass B(y,x);

A.Fun(B);

```

Question 7. Given the main program and the 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) **(5 PTS)** What are the contents of p and q after the execution of line (X)?

SOLUTION:

a) Since call by value is performed the contents of p and q are unchanged.

b) **(5 PTS)** 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) .

SOLUTION:

b)

1st method:

```
void Fun(int &*a, int &*b)
{
    a = a - 3;
    b = b - 5;
}
```

In line (X):

```
Fun(p,q); // Line (X)
```

2nd method:

```
void Fun(int **a, int **b)
{
    *a = *a - 3;
    *b = *b - 5;
}
```

In line (X):

```
Fun(&p,&q); // Line (X)
```

Question 8. (20 pts) Complete the output of the code given below.

<pre> 01 #include <iostream.h> 02 #include <stdlib.h> 03 04 int func1(int q[], int n=4, int m=0) 05 { 06 int j; 07 for(j=0; j<n; j++) cout<<q[j]+m<<" "; 08 cout<<endl; 09 return *(q+m+n+1); 10 } 11 12 int* func2(int *q[], int n=4, int m=0) 13 { 14 int j; 15 for(j=0; j<n; j++) cout<<*(q[j]+m)<<" "; 16 cout<<endl; 17 return *(q+m+n+1); 18 } 19 20 21 int** func3(int **q[],int n=4, int m=0) 22 { 23 int j; 24 for(j=0; j<n; j++) cout<<** (q[j]+m)<<" "; 25 cout<<endl; 26 return *(q+m+n+1); 27 } 28 29 void main() 30 { 31 int i, a, A[100]; 32 int *p, *P[100], **r, **R[100]; 33 for(i=0; i<100; i++) 34 { A[i]=i+i; 35 P[i]=&A[i]+i+1; 36 R[i]=&P[i]+i+1; 37 } 38 39 cout<<endl<<"line 40:"<< endl; 40 a=func1 (A,5,3); cout<<"out:"<<a<<endl; 41 cout<<endl<<"line 42:"<< endl; 42 p=func2 (P,4,2); cout<<"out:"<<*p<<endl; 43 cout<<endl<<"line 44:"<< endl; 44 r=func3 (R,4,2); cout<<"out:"<<**r<<endl; 45 cout<<endl<<"line 46:"<< endl; 46 a=func1 (A+*P[2]); 47 cout<<endl<<"line 48:"<< endl; 48 func1 (A+* (*R[1]+3)); 49 cout<<endl<<"line 50:"<< endl; 50 func1 (func2 (R[3])); 51 cout<<endl<<"line 52:"<< endl; 52 func1 (func2 (func3 (R+1))); 53 } </pre>	<p>OUTPUT:</p> <p>line 40: 3,5,7,9,11, out: 18</p> <p>line 42: 6,10,14,18, out: 30</p> <p>line 44: 14,22,30,38, out: 62</p> <p>line 46: 20,22,24,26,</p> <p>line 48: 40,42,44,46,</p> <p>line 50: 30,34,38,42, 50,52,54,56,</p> <p>line 52: 14,22,30,38, 54,58,62,66, 74,76,78,80,</p>
---	--

STACKS QUEUES

Question 1) (25 pts)

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

```
const int MaxStackSize=50;
template <class T>
class Stack
{
private:
T stacklist[MaxStackSize];
int top;
public:
Stack(void); // constructor
to initialize top
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;
};
```

Part a) 12 points

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());
}
```

Part b) 13 points You are given the following piece of code. Write OUTPUT1 and OUTPUT2 in the respective boxes.

```
#include "stack.h"
#include "queue.h"
void main ()
{
    int i;
    Queue<int> Q1,Q2;
    Stack<int> S1,S2;
    int Arr[6];
    for(i=0;i<5;i++)
    {Q1.QInsert(i);
    Q2.QInsert(i*2);}
    i=0;
    while(!Q1.QEmpty())
    {
        Arr[i]=Q1.QDelete();
        if(Arr[i]<3)
            Arr[i]=Arr[i]*2;

        S1.Push(Arr[i]);
        S2.Push(Q2.QDelete());

        if(Arr[i]>3)
            Arr[i]=S1.Pop()+S2.Pop();
        i++;
    }
}
```

```

    }
    while (!S1.StackEmpty())
        S2.Push(S1.Pop());

    //OUTPUT1
    cout<<"stack:";
    while (!S2.StackEmpty())
        cout<<S2.Pop()<<" ";
    cout<<"\n";
    //OUTPUT2
    cout<<"array:";
    for(i=0;i<5;i++)
        cout<<Arr[i]<<" ";
}
```

OUTPUT1:

stack:0 2 3 6 2 0

OUTPUT2:

array:0 2 8 3 12

Question 2 (25 pts.) You are given the following `Stack` class definition, implemented as discussed in class:

```
MaxSize = 50; //constant capacity
template <class T>
class Stack
{private:
    T slist [MaxSize];
    int top;
public:
    Stack(void);
    void Push (const T &item);
    T Pop (void);
    int Stack_Empty (void) const;
    int Stack_Full (void) const;
}
```

Now, you are required to implement a `Queue` class using ONLY stacks (i.e., objects that belong to the class given above,) to store the data items. That is, you can NOT use an array to store the items in the queue, you have to use a stack for that purpose.

(a) (5 pts.) Complete the private part of the following template-based `Queue` class definition:

```
template <class T>
class Queue
{private:
    .....
public:
    Queue (void);
    void QInsert (const T &item);
    T QRemove (void);
    int QEmpty (void);
    int QFull (void);
}
```

(b) (15 pts.) Give the implementation of the `QInsert` and `QRemove` member functions of the `Queue` class, to insert an item at the rear of a queue and to remove the item at the front of the queue, respectively.

(c) (5 pts.) Give the implementations of the `QEmpty` and `QFull` member functions of the `Queue` class, to test whether the queue is empty or full, respectively.

SOLUTION

(a)

```
template <class T>
class Queue
{private:
    Stack<T> SLIFO, SFIFO;
public:
    Queue (void);
    void QInsert (const T &item);
    T QRemove (void);
    int QEmpty (void);
    int QFull (void);
}
```

(b) Solution idea: Insert items into the LIFO stack; whenever a removal is required (i.e. QRemove called) simply reverse the LIFO into another stack called FIFO, remove from the top, and then restore the LIFO stack.)

```
void QInsert (const T &item)
{if SLIFO.Stack_Full()
    {cerr<<"Queue overflow"<<endl; exit(1);}
  Push.SLIFO(item); //insert directly to SLIFO
}

T QRemove (void)
{T temp;
  if SLIFO.Stack_Empty()
    {cerr<<"Queue empty"<<endl; exit(1);}
  while (!SLIFO.Stack_Empty( )) //reverse order
    SFIFO.Push(SLIFO.Pop());
  temp = SFIFO.Pop( ); //remove the first in item
  while (!SFIFO.Stack_Empty())
    SLIFO.Push(SFIFO.Pop()); //restore order
  return temp;
}
```

(c)

```
int QEmpty (void)
return SLIFO.Stack_Empty();

int QFull (void)
return SLIFO.Stack_Full();
```

Question 3. (25 pts.)

Write a C++ function, `unbalanced_string`, which checks for balancing of, parenthesis ('(', ')'), brackets ('[', ']') and braces ('{', '}') in a given string of arbitrary characters. For a string to be balanced, every right brace, bracket and parenthesis must correspond to its left counterpart. For example, the sequence `[()]` is balanced but `[()]` is not. The function

```
int unbalanced_string(char *s)
```

will return the result `-1` if the string is balanced or the "index" of the first unmatched left character '(', '[' or '{' in the string if it is unbalanced. In case of a missing left character, the index of the first unmatched right character will be returned. You are allowed to use only the stack data structure developed in the class for this purpose. Two examples of balanced and unbalanced input strings are `{x(y[wz])}` and `{x(y[wz])}`, for which, `unbalanced_string` returns `-1` and `3`, respectively.

SOLUTION:

```
int unbalanced_string(char *s)
{ Stack<char> Sc;
  Stack<int> S1, S2, S3;
  char c; int i=0; int x,y,z;
  while (*s != NULL)
  { if (*s == '(') {Sc.Push(*s); S1.Push(i)}
    else if (*s == '[') {Sc.Push(*s); S2.Push(i)}
    else if (*s == '{') {Sc.Push(*s); S3.Push(i)}
    /* else statements in the following three blocks check
       the invalidity case such as "[()]" */
    else if (*s == ')') {
      if S1.Empty() return i
      else {c=Sc.Pop();
            x=S1.Pop();
            if (c=='(')
              else return x}
    }
    else if (*s == ']') {
      if S2.Empty() return i
      else {c=Sc.Pop();
            x=S2.Pop();
            if (c=='[')
              else return x}
    }
    else if (*s == '}') {
      if S3.Empty() return i
      else {c=Sc.Pop();
            x=S3.Pop();
            if (c=='{')
              else return x}
    }
  }
  s++; i++;
}
/* now check unbalanced case. If so, return the index of the
   first unmatched character */
if (S1.Empty() && S2.Empty() && S3.Empty()) return -1
else {
  x = y = z = Infinity; /* a very large number */
  while !S1.Empty() x = S1.Pop();
  while !S2.Empty() y = S2.Pop();
  while !S3.Empty() z = S3.Pop();
  /* find the minimum index */
  if (x<=y && x<=z) return x
  else if (y<=x && y<=z) return y
  else if (z<=x && z<=y) return z
  else cout << "error"
}
}
```

Question 5. (25 pts) The following Stack and Queue classes are given:

STACK CLASS	QUEUE CLASS
<pre>const int MaxStackSize=50; template <class T> Class stack {private: int top; T stacklist [MaxStackSize]; public: Stack(void);//constructor //stack modification operations void Push(const T& item); //insert at top T Pop(void); // delete from top void SClear(void); //stack access T Peek(void) const; // return top item // stack test methods int SEmpty(void) const; int SFull(void) const; };</pre>	<pre>const int MaxQSize=50; template <class T> class Queue {private: int front, rear, count; T Qlist [MaxQSize]; public: Queue(void); //constructor //queue modification operations void QInsert(const T& item); // insert at rear T QDelete(void); // delete from front void QClear (void); // queue access T QFront(void) const; // return front item // queue test methods int QLength(void) const; int QEmpty(void) const; int QFull(void) const; };</pre>

a) (14 pts) Complete the implementations of the following two functions which are not members of the given classes (i.e. these are NOT member functions but global functions):

```
template <class T>
T RetrieveElement (Stack<T> &S, int i)
/* returns the i'th element counted from the top of Stack S,
   return value is undefined if i is out of bounds of stack */
```

```
{Stack<T> temp; T tempel;// temporary
  storage
  for (int j=0; !S.EMpty() && j<>i; j++)
    temp.Push(S.Pop());
  if j == i tempel = temp.Peek( );
  while (!temp.SEMpty( ))
    S.Push(temp.Pop());
  return tempel;
}
```

Part (a) continued on reverse →

→ Question 5. part (a) continued:

```
template <class T>
void StoreElement(Stack<T>& S, int i, const T& item)
/*stores given item i at the i'th location from stack top,
  undefined items of type T pushed into stack at previously
  unused locations
*/
{
    T UNDEF; //contents undefined
    Stack<T> temp; // temporary storage
    for (int j=0; !S.Empty() && j<i; j++)
        temp.Push(S.Pop());
    while (j<i) { temp.Push(UNDEF); j++; }
    //at this point j=i
    S.Push(item);
    while (!temp.Empty())
        S.Push(temp.Pop());
    return;
}
```

b) (8 pts) Complete the implementation of the following function to insert an element, not at the rear, but at the front of a given queue, using only the public methods of the given queue class, and an extra queue for temporary storage:

```
template <class T>
void QInsertFront (Queue<T>& Q, const T& item)
{
    Queue<T> temp; //temporary storage
    while (!Q.QEmpty())
        temp.QInsert(Q.QDelete());
    Q.QInsert(item); //store item at final place
    while (!temp.QEmpty()) //restore original
        Q.QInsert(temp.QDelete());
}
```

c) (3 pts) Write the $O(\cdot)$ complexity of the operations in part(a) and in part(b), if the original stack and queue contains n items:

(a) RetrieveElement:	StoreElement:	(b)Q InsertFront:
$O(n)$	$O(n)$	$O(n)$

Question 6.

(4 pts) You are given the following piece of code. Write OUTPUT1 and OUTPUT2 in the respective boxes.

```
#include "stack.h"
#include "queue.h"
void main ()
{
    int i;
    Queue<int> Q1,Q2;
    Stack<int> S1,S2;
    int Arr[5];
    for(i=0;i<5;i++)
    {Q1.QInsert(i);
    Q2.QInsert(i*2);}
    i=0;
    while(!Q1.QEmpty())
    {
        Arr[i]=Q1.QDelete();
        if(Arr[i]<3)
        Arr[i]=Arr[i]*2;

        S1.Push(Arr[i]);
        S2.Push(Q2.QDelete());

        if(Arr[i]>3)
        Arr[i]=S1.Pop()+S2.Pop();
        i++;
    }
    while(!S1.StackEmpty())
    S2.Push(S1.Pop());

    //OUTPUT1
    cout<<"stack:";
    while(!S2.StackEmpty())
    cout<<S2.Pop()<<" ";
    cout<<"\n";
    //OUTPUT2
    cout<<"array:";
    for(i=0;i<5;i++)
    cout<<Arr[i]<<" ";
}
```

OUTPUT1:

stack:0 2 3 6 2 0

OUTPUT2:

array:0 2 8 3 12

Question 8

You are given the Stack and Queue Class Declarations:

<pre>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; }</pre>	<pre>template <class DataType> class Queue { private: int front, rear, count; DataType qlist[MaxQSize]; public: Queue(void); Qinsert(const Datatype item); DataType Qdelete(void); void ClearQueue(void); DataType QFront(void) const; int QLength(void) const; int QEmpty(void) const; int Qfull(void) const; }</pre>
--	--

Notice that all the data members in these classes private, that is they can not be accessed outside the classes. You are required to write the function SwapStack which will swap the content of the top and bottom elements of a stack (by following steps a and b below). An example case before and after this fuction is called is given below.

Before SwapStack		After SwapStack	
top		top	
	10		127
	23		23
	7		7
	12		12
bottom	127	bottom	10

a) Complete the following code so that the SwapStack function will operate properly. You are not allowed to declare any other data structures but only those given in the fuction below.

```

Template <Class T>
void SwapStack(Stack<T> myS)
{ T topItem, bottomitem;
  Queue<T> myQ;
  // get topItem from the Stack
  topItem=myS.Pop();
  // move the rest of the myS into myQ
  while ...
  { ...
    };
  // load the bottomItem appropriately
  bottomItem=...
  // other operation(s)between myS and myQ if any necessary

  ...

  // push  either topItem or bottomItem into myS
  // whichever appropriate
  ...
  // push other items into myS by taking from myQ
  while ...
  { ...
    };
  // push  either topItem or bottomItem into myS
  // whichever appropriate
  ...
}

```

b) Now assume that the data member top and stacklist are also public and repeat (a)

```

Template <Class T>
void SwapStack(Stack<T> myS)
{ T ...
  topitem= ...

  ...

  ...                      =topitem;
}

```

SOLUTION: You are given the Stack and Queue Class Declarations:

<pre> 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; } </pre>	<pre> template <class DataType> class Queue { private: int front, rear, count DataType qlist[MaxQSize]; public: Queue(void); Qinsert(const Datatype item); DataType Qdelete(void); void ClearQueue(void); DataType QFront(void) const; int QLength(void) const; int QEmpty(void) const; int Qfull(void) const; } </pre>
--	---

Notice that all the data members in these classes private, that is they can not be accessed outside the classes. You are required to write the function SwapStack which will swap the content of the top and bottom elements of a stack.

a) Complete the following code so that the SwapStack function will operate properly. You are not allowed to declare any other data structures but only those given in the fuction below.

```

Template <Class T>
void SwapStack(Stack<T> myS)
{ T topItem, bottomitem;
  Queue<T> myQ;
  // get topItem from the Stack
  topItem=myS.Pop();
  // move the rest of the myS into myQ
  while !(myS.StackEmpty())
  { myQ.Qinsert(myS.Pop()); }
  // load the bottomitem appropriately
  bottomItem=myQ.Qdelete();
  // solutions setting Bottom item previously or later
  // also accepted
  //
  // write here other operation(s)between myS and myQ
  // if there is any that you find necessary
  while !(myQ.QEmpty())
  { myS.push(bottomItem);bottomItem=myQ.Qdelete(); }
  while !(myS.StackEmpty())
  { myQ.Qinsert(myS.Pop()); };
  // push topItem into myS
  myS.Push(topItem);
  // push other items into myS by taking from myQ
  while !(myQ.QEmpty)
  myS.Push(myQ.Qdelete());
  // push bottomItem into myS
  myS.Push(bottomItem);
}

```

b) Now assume that the data member top and stacklist are also public and repeat (a)

```

Template <Class T>
void SwapStack(Stack<T> myS)
{ T topitem;
  topitem=myS.Peek();
  myS.stacklist[top]= myS.stacklist[0];
  myS.stacklist[0]= topitem;
}

```


COMPLEXITY

Question 1 (25 pts)

a) Given the following functions covered in class:

```
template <class T>
int SequentialSearch (T list[ ], T key, int N);
template <class T>
int BinarySearch (T list[ ], int low, int high, T key);
and the following function which returns a random integer with O(1) complexity.
int RandInt ();
```

Function RandomSearch performs a search to find a given key in a sorted array of N items.

RandomSearch is defined as follows:

```
template <class T>
int RandomSearch (T list[ ], int low, int high, T key, int N)
{
    int index;
    int x=RandInt();
    if(x%2==0)//x mod 2 operation
        index= SequentialSearch (list[ ], key, N);
    else
        index=BinarySearch (list[ ], 0, N-1, key);
    return index;
}
```

Find the complexity of RandomSearch in $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$. Justify your results. If you cannot express the complexity in any of these forms, state your reasons.

Sequential Search : $O(N)$, $\Omega(1)$ Binary Search : $O(\log N)$, $\Omega(1)$ Random Search: $O(N)$, $\Omega(1)$, No $\Theta(\cdot)$.

b) Consider a two dimensional array that holds N integers. The array has a fixed column size of K . For each row i , K random integers are picked between \min_i and \max_i such that $\min_i > \max_{i-1}$ and $\max_i < \min_{i+1}$ and stored in the array.

You are given the following algorithm to search for a given integer key for such a two dimensional array:

For each row i , the algorithm checks if the key is between \min_i and \max_i . If that is the case, then it compares the key with all items in row i until it finds a match. If there is no match it exits without a match.

Find the complexity of this search algorithm in $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$. Justify your results. If you cannot express the complexity in any of these forms, state your reasons.

Number of rows in the array= N/K N/K comparisons to find the row and then K comparisons to find the item in the correct row. K is a constant. Complexity= $O(N/K+K)$ Complexity= $O(N)$ $\Omega(1)$, No $\Theta(\cdot)$.

Question 2

Determine, showing your arguments briefly, the $O()$, $\Omega()$ and $\Theta()$ complexities of:

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

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

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

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

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

Solution:

(a) $f(n) = n^2 - 1 + (8/(n+1))$ so $f(n) = O(n^2)$, $\Omega(n^2)$ and $\Theta(n^2)$

(b) The inner loop is executed

$1^2 + 2^2 + 3^2 + \dots + n^2$ times, so

$f(n) = n(n+1)(2n+1)/6 = O(n^3)$, $\Omega(n^3)$ and $\Theta(n^3)$

(c) In each step, the table of n items is partitioned into 3

This can be done at most $\log_3 n$ times.

The exit operation is of constant complexity

(at most three comparisons, regardless of total table size,)

So, total worst case complexity is $O(\log n)$.

(Note: "best" case is of course $\Omega(1)$ corresponding to success at the first comparison, hence $\Theta()$ is not defined.)

Question 3. (30 pts) Show all your reasoning and work.

a) (6 pts) Let $f(n) = n$, $g(n) = n^{(1+\sin(n))}$ and $h(n) = n^2$.

Is $f(n) = O(g(n))$ correct?

Is $f(n) = \Omega(g(n))$ correct?

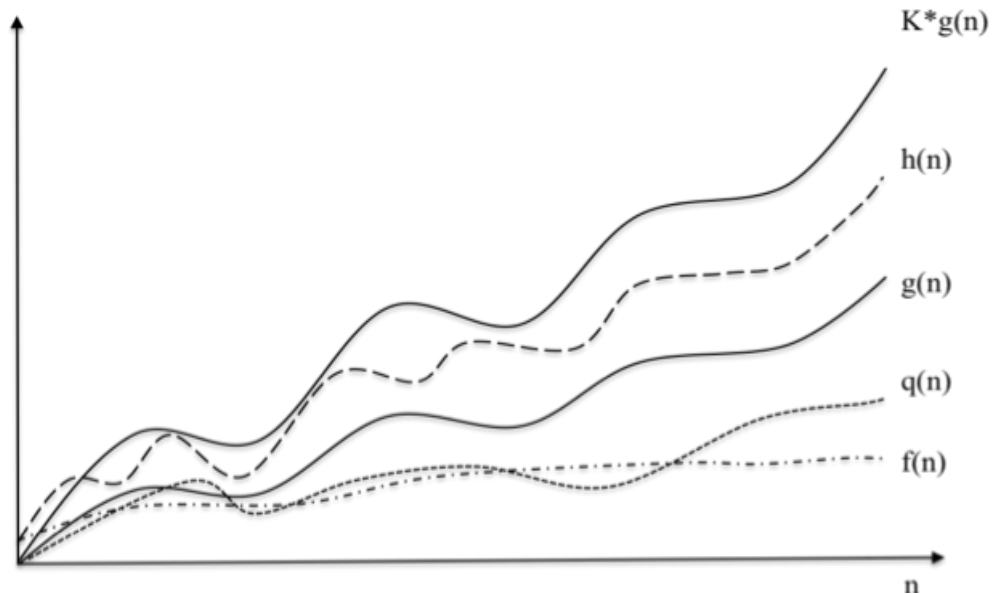
Is $h(n) = \Omega(g(n))$ correct?

Explain.

Solution:

First two statements are incorrect since $g(n) = n^{(1+\sin(n))}$ changes between 1 and n^2 .
The third statement is correct.

b) (6 pts) Consider the given figure. Let $p(n) = h(n) + f(n)$



Find the TIGHTEST $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$ complexities of $h(n)$, $q(n)$ and $p(n)$ expressed in terms of $f(n)$ and $g(n)$.

Solution:

$h(n)$: $\Theta(g(n))$

$q(n)$: $O(g(n))$, $\Omega(f(n))$

$p(n)$: $\Theta(g(n))$

c) (18 pts) Consider two arrays of integers A1 and A2. Both A1 and A2 store N distinct integers where $N=2^k-1$, k is an integer. The contents of A1 and A2 are independent of each other. A2 is sorted from smallest to largest. You are required to find how many integers in A1 also exist in A2 using the binary search algorithm covered in the lectures.

What is the complexity of this task in terms of the TIGHTEST $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$?

Hint1: Consider the best and worst cases. *Construct an example for the best case.*

Hint2: $\sum_{i=1}^m i2^i = 2(1 + (m-1)2^m)$

Answer:

Complexity in $O(\cdot)$: $M\log N$

Complexity in $\Omega(\cdot)$: $M\log N$

Complexity in $\Theta(\cdot)$: $M\log N$

Solution:

Worst case: N unsuccessful searches, each terminates in $\log N$. Complexity in $O(\cdot)$: $M\log N$

Best case: All N items are found with the shortest number of trials. Consider the following example:

A1=[4,2,6,1,3,5,7], A2=[1,2,3,4,5,6,7]

Search for 4: 1 cut in the middle, found.

Search for 2 and 6: 1 cut in the middle, not found, 1 more cut, found.

Search for 1,3,5,7: found after 3 cuts.

You find 1 element with 1 cut, 2 elements with 2 cuts, 4 elements by 3 cuts. Generalize: You find i elements with 2^{i-1} cuts.

You need a total of $\sum_{i=1}^k i2^{i-1} = (1 + (k-1)2^k)$ operations for $N=2^k-1$.

Complexity in $\Omega(\cdot)$: $M\log N$

Complexity in $\Theta(\cdot)$: $M\log N$

Question 4. (25 pts) Solutions that only consist of expressions $O()$, $\Omega()$ and $\Theta()$ will receive no credit. Show all your reasoning and work.

a) Derive the complexity of the following code segment in $O()$, $\Omega()$ and $\Theta()$:

```
int A[N];
int B[N];
bool found=false;
int i,j;
int key;
cin>>key;//get the key from the user
for(i=0;i<N;i++)
for(j=0;j<N;j++)
if(A[i]==B[j])
{
found=true;
break;//breaks both for loops
}

if(found==true)
BinarySearch (A, 0, N-1, key);
```

Recall: BinarySearch (int list[], int low, int high, int key)

Worst case: Match is found for $A[N-1]==B[N-1]$: $O(N^2)$

Then conduct binary search $O(\log N)$.

Total: $O(N^2) + O(\log N) = O(N^2)$

Best case Match is found for $A[0]==B[0]$ then conduct Binary search: $\Omega(1)$ if key found at the first checked location.

$\Theta()$ does not exist.

Note if no match is found, $O(N^2)$ but this is not the complexity

b) Consider a stack with N items.

What is the $O()$ complexity of accessing the most recently inserted item in the stack.

What is the $O()$ complexity of accessing oldest item in the stack.

$O(1)$: it is the top item

$O(N)$: it is the bottom item, all other items have to be taken out first

c) What is the $O()$ complexity of searching for a given key in a stack or a queue ?

$O(N)$: Both of these data structures provide sequential access.

Question 5 (25 pts)

a) Given the following function:

```
void DoSomething (int N);
{
  int i=0;
  int j=0;
  while(i<N)
  {
    statement x;
    while(i<N)
    {i++;
     statement y;}
    j++;
    statement z;
  }
}
```

The complexity of statements are given as:

statement x: $O(\log N)$

statement y: $O(N)$

statement z: $O(1)$

What is the complexity of DoSomething in $O(\cdot)$?

b) You are given the function

int Diff(int i, int j) which returns the absolute value of i-j.

Consider the function FindMinDiff below which takes an array A of size N and returns the minimum absolute difference between any two integers in A.

```
int FindMinDiff ( int A[], int N)
{
  int i,j,diff;
  int MinDiff=Diff(A[0],A[1]);
  for(i=0;i<N;i++)
  for(j=i+1;j<N;j++)
  {
    diff=Diff(A[i],A[j]);
    if(diff<MinDiff)
    MinDiff=diff;
  }
  return MinDiff;
}
```

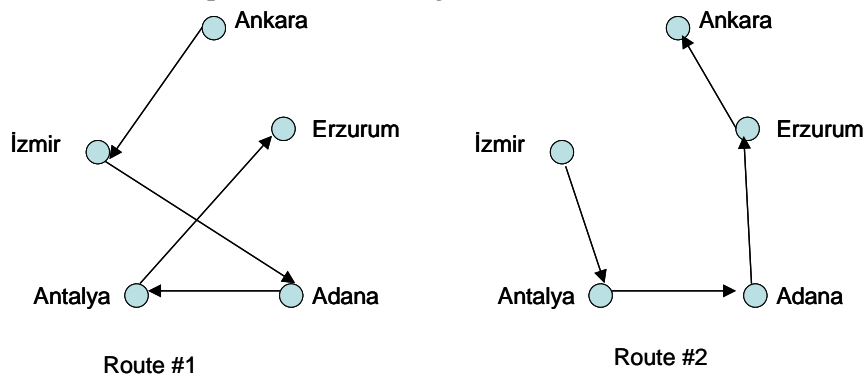
What is the complexity of FindMinDiff in $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$?

c)Traveling Salesman Problem:

A salesman wants to visit N towns to sell books. He visits every town exactly once. He can start from any town.

Example:

For 5 towns, two possible routes are given below.



Let $F(N)$ be the function that computes the route with the shortest possible route for the traveling salesman. What is the complexity of $F(N)$ in $O(.)$ notation.

Solution:

a)

```
void DoSomething (int N);
{
    int i=0; → ti
    int j=0;
    while(i<N) → ta
    {
        statement x; → tx
        while(i<N) → ta
        {i++; → tb
        statement y;} → ty
        j++; → tc
        statement z; → tz
    }
}
```

Complexity: $t_i + t_a + t_x + N(t_a + t_b + t_y) + t_c + t_z = t_A + t_x + N(t_B + t_y) + t_z \rightarrow$
 $t_A + O(\log N) + N(t_B + O(N)) + O(1) = t_C + O(\log N) + N t_B + N O(N) = O(N^2)$

b)

```
int FindMinDiff ( int A[], int N)
{
    int i,j,diff; → ti
    int MinDiff=Diff(A[0],A[1]); → tD
    for(i=0;i<N;i++) → ta1,ta2,ta3
    for(j=i+1;j<N;j++) → tb1,tb2,tb3
```

```

{
diff=Diff(A[i],A[j]); ➔ tD
if(diff<MinDiff) ➔ tc
MinDiff=diff; ➔ te
}
return MinDiff; ➔ tr
}

```

```

for (j=i+1;j<N;j++) ➔ tb1, tb2, tb3
==
for (j=0;j<N-i-1;j++) ➔ tb1, tb2, tb3

```

$t_i + t_D + t_{a1} + (N+1)t_{a2} + Nt_{a3} + \sum_{i=1}^N (tb_1 + (N-i)tb_2 + (N-i-1)ta_3)$
 $O(N^2)$ $\Omega(N^2)$, $\Theta(N^2)$

c)
Brute force search: Check all combinations of cities: $O(N!)$

Question 6) (25 pts)

a) Given the following function:

```
void JustForComplexity (int N, int k)
{
    T1(N);
    for (int i=0; i<N; i++)
        if (k<N)
            T2(N);
        else
            {
                T3(N);
                return;
            }
}
```

The complexity of T1, T2 and T3 are given as:

T1(N) : $\Theta(N)$

T2(N) : $O(N \log N)$

T3(N) : $\Theta(N^2)$

Derive the complexity of JustForComplexity in $O(\cdot)$

b) 4 algorithms A, B, C and D must be executed on a set of N items to produce a certain result R. At time $t=0$ algorithms A, B and C start to run in parallel and finish at times t_A , t_B and t_C respectively. R is correct only when $t_A < t_B < t_C$ order is satisfied. Algorithm D starts to run when algorithm C is finished.

The known information about the complexities of the algorithms is as follows:

Algorithm A: $\Theta(N)$

Algorithm C: $\Theta(N^3)$

Algorithm D: $O(2^N)$

i) What should be the complexity of Algorithm B in $O(\cdot)$ and $\Omega(\cdot)$ such that R is correct.

ii) What is the overall complexity of producing the correct result in $O(\cdot)$.

Solutions that only consist of expressions $O(\cdot)$, $\Omega(\cdot)$ will receive no credit. Also explain your reasoning in deriving them

Solution:

a) if ($k < N$): T1(N) and $N \cdot T2(N)$ will be executed.
 $O(N) + N \cdot O(N \log N) \Rightarrow O(N^2 \log N)$ is the dominant term
 if ($k \geq N$): T1(N) and T3(N) will be executed.
 $O(N) + O(N^2) \Rightarrow O(N^2)$ is the dominant term
 Over all: $\max(O(N^2 \log N), O(N^2)) = O(N^2 \log N)$

b)

i) Algorithm B must be finished after A. So the shortest execution time of B must be longer than the longest execution time of A. Algorithm B must be $\Omega(f(N))$ such that $f(N) > N$ as N goes to infinity. One possible choice is $\Omega(N^2)$. Algorithm B must be finished before C. So the longest execution time of B must be shorter than C. Algorithm B must be $O(g(N))$ such that $g(N) < N^3$ as N goes to infinity. One possible choice is $O(N^2)$.

ii) $O(2^N)$

Examples on Dynamic Memory Management

Q1)

```
template <class T>
class MM
{
private:
    T x;
    MM<T>* m;
    T* t;
public:
    MM(T Xin, MM<T>* Min=NULL)
    {
        x=Xin;
        m=Min;
        if(Min==NULL)
            t=new T(Xin);
        else
            t=new T(Min->x);
    };
    MM(const MM<T>& MMin);
    ~MM();
}
```

(a) Draw a diagram that shows the data structures created by the following statements.

```
MM<int> p(3), *q;
```

```
q = new MM<int>(5,&p);
```

(b) Implement the copy constructor and destructor.

Q2)

```
1- #include <iostream.h>
2-
3- class MyClass{
4-     private:
5-         char *c;
6-     public:
7-         MyClass(const int& n);
8-         char& Put(const int n);
9-         char Get(const int n);
10-    };
11-
12- MyClass::MyClass(const int& n)
13-    { c= new char[n] };
14-
15- char& MyClass::Put(const int& n)
16-    { return c[n]; };
17-
18- char& MyClass::Get(const int& n)
19-    { return c[n]; };
20-
21- void MyFn(MyClass& m1)
22-    {
23-        MyClass *mc;
24-
25-        mc = new MyClass(10);
26-        mc->Put(3) = 'a';
27-        m1.Put(3) = mc->Get(3);
28-    };
29-
30- main()
31- {
32-     int n;
33-     MyClass m1(20);
34-
35-     cin >> n;
36-     MyFn(m1);
37-     cout << m1.Get(3);
38- };
```

What is the major programming error related to dynamic memory usage in this program?
Indicate how you would correct this error by giving the line numbers of the lines you would delete, if necessary, and adding new code, if necessary.

Q3) Consider the following C++ class declaration:

```
class Z
{
private:
    int *z1; int *z2;
public:
    void Z(const int x1, x2);
    void Z(const Z &x);
    int *first (void) {return z1};
    int *second (void) {return z2};
}
```

(a) Assuming that a complete implementation of this class, exactly as it is declared here, is available, draw the constructed data structures after the following program sequence is executed:

```
...
Z *zp;
zp = new Z(3,5);
Z a(6, *(zp->first() ) ), b=a, c(0,0);
c = *zp;
delete zp;
```

(b) Give an appropriate implementation for the destructor function for this class.

Q4) Consider the following C++ class declaration:

```
template <class T>
class MT2
{
private: int n; T *p;
public: MT2(const T &m1, int nn=0)
    {
        n=nn;
        if (n>0) p= new T[n];
        for (int i=0; i<n; i++) *(p+i) =m1;
    };
    MT2(const MT2<T> &m);
    ~MT2(void);
    MT2<T> &operator= (const MT2<T> &mt2obj);
}
```

a) Draw a diagram that show the data structures created by the following C++ statements:

```
MT2<int> A(1), *q;
q=new MT2(7,5);
```

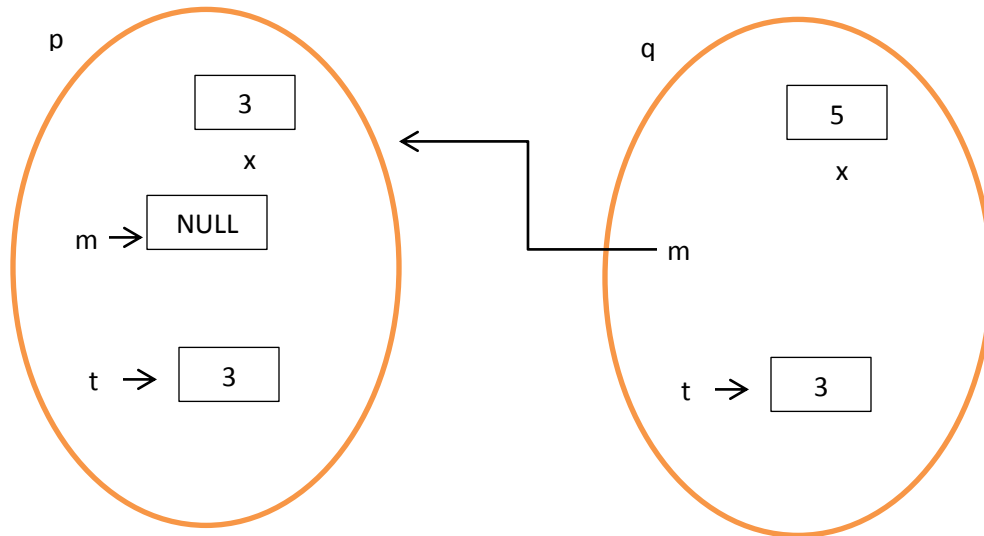
b) Implement the copy constructor and destructor functions of this class.

c) Implement the overloaded assignment operator for this class so that, regardless of the original contents of the left hand side, after assignment, it will be a copy of the right hand side.

Solutions:

Q1)

(a)



(b)

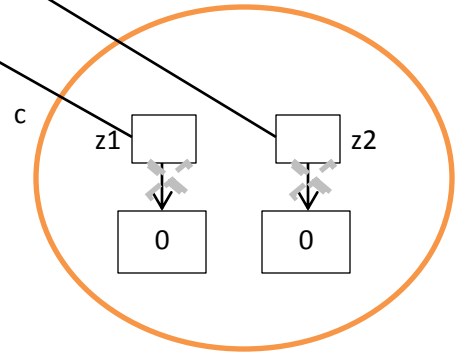
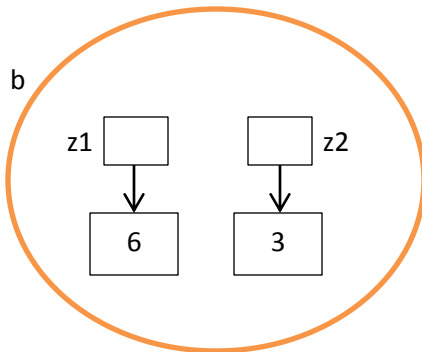
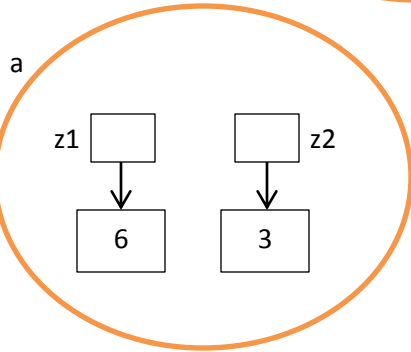
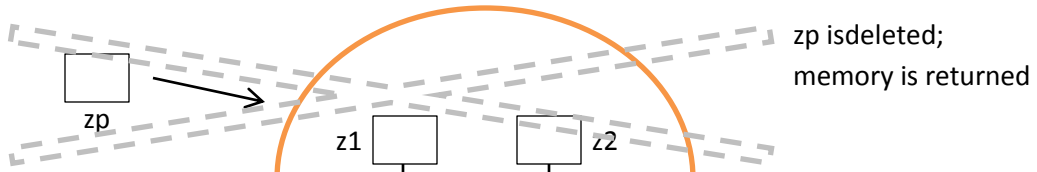
```
template <class T>
MM::MM(const MM<T>& MMin)
{
    x=MMin.x;
    m=MMin.m;
    t=new T(*MMin.t);
};

template <class T>
~MM(void)
{
    delete t;
};
```

Q2) Destructor is missing because the class uses dynamic memory to allocate memory space. This space must be returned to the system memory manager.

```
3- class MyClass{
4-     private:
5-         char *c;
6-     public:
7-         MyClass(const int& n);
8-         char& Put(const int n);
9-         char Get(const int n);
10-         ~MyClass();
11- };
12- MyClass::~~MyClass()
13- { delete []c; };
```

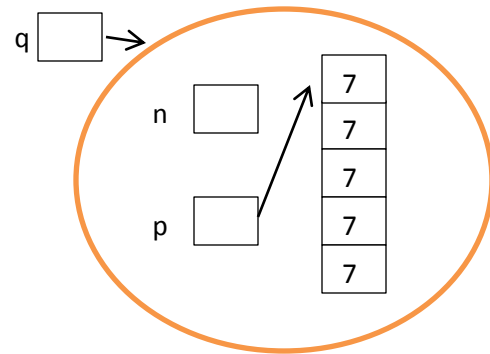
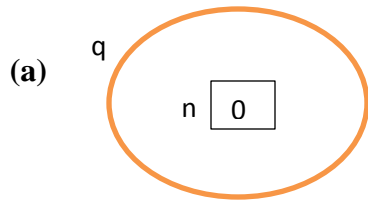
Q3)
(a)



(b)

```
Z::~~Z(void)
{ delete z1; delete z2; }
```

Q4)



(b)

```
template <class T>
MT2<T>::MT2(const MT2<T> &m)
{
    n=m.n;
    if (n) p=new T[n];
    for (int i=0; i<n; i++)
        *(p+i)=*(m.p+i);
}
```

```
template <class T>
MT2<T>::~~MT2(void)
{ if(n) delete p[n]; }
```

(c)

```
template <class T>
MT2<T>&MT2<T>::operator=(const MT2<T> &mt2obj)
{
    if (n) delete p[n];
    n=mt2obj.n;
    if (n) p=new T[n];
    for (int i=0; i<n; i++)
        *(p+i)=*(mt2obj.p+i);
    return *this;
}
```

LINKED LIST

Question 1)

You are given a modified version of the Node Class covered in the lectures in which access to the next pointer is public.

```
template <class T>
class Node
{
    public:
        T data;
        Node<T> *next;

        Node (const T& item, Node<T>* ptrnext = NULL);

        void InsertAfter(Node<T> *p);
        Node<T> *DeleteAfter(void);

        Node<T> *NextNode(void) const;
};
```

Consider the SwapNodes global function template is given as follows:

```
template <class T>
void SwapNodes(Node<T>* head, T data1, T data2)
```

SwapNodes function:

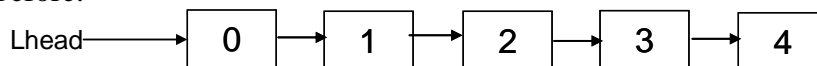
- Takes a linked list pointed by the head pointer. **All of the nodes of this linked list were created dynamically before.**
- Swaps the nodes whose data fields have values of data1 and data2.

Assume that:

- It is guaranteed that the nodes whose data fields have values of data1 and data2 exist in the list
- data1 and data2 appear only once in the list
- Head node never contains data1 or data2

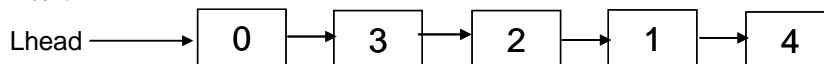
Example:

Before:



SwapNodes (Lhead, 1, 3);

After:



Part a) Implement the SwapNodes function. You must use only the existing nodes of the list. **Do not create any new nodes in your implementation even for temporary use.**

Part b) What is the complexity of the SwapNodes function in

$O()$

$\Omega()$

$\Theta()$

Justify your answers.

Solution (Compiled and works):

Part a)

```

template <class T>
void SwapNodes(Node<T>* head, T data1, T data2)
{
Node<int> * p , *q , *pA, *pB, *temp;

p=head;
//find node with data1
while (p->next->data!=data1)
p=p->next;
pA=p->next;

q=head;
//find node with data2
while (q->next->data!=data2)
q=q->next;
pB=q->next;

//update pointers and swap the nodes
p->next=pB;
q->next=pA;
temp=pA->next;
pA->next=pB->next;
pB->next=temp;

}

```

WONT WORK:

```

p->next=pB;
temp=pA->next;
q->next=pA;
pA->next=pB->next;
pB->next=temp;

```

```

temp=pA->next;
p->next=pB;
q->next=pA;
pA->next=pB->next;
pB->next=temp;

```

Part b) $O(N)$, N is the size of the list. We have to scan the list to find the items. $\Omega(1)$ No Θ

Question 2) For this question, use the following modified Node class definition:

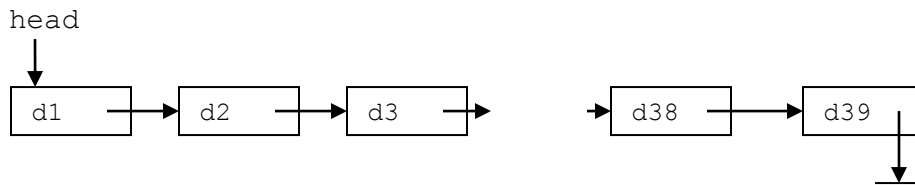
```
template <T>
```

```
class Node {public: T data; Node<T> *link; Node (T d, Node<T> p);}
```

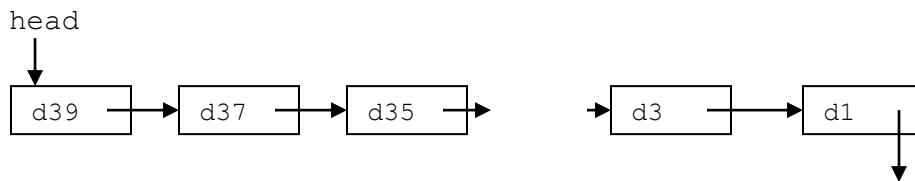
a) (5 pts.) Complete the InsertFront () function that will insert the node pointed by p to the front of the list whose first node is pointed by head.

```
template <class T>
void InsertFront (Node<T> *&head, Node<T> *p)
{if (p==0) return;
 else
     { ___ ; //link new node to list
       ___ ;} //link head to new node
}
```

b) (20 pts.) Now, complete the function Rev_odd () using InsertFront (). Rev_odd will return with head pointing to a new list whose items consist of only the items positioned at the odd numbered nodes of the input list, in reverse order. That is, for the following list:



The call Rev_odd(head); returns with:



```
template <class T>
void Rev_odd (___ head) //pointer to list

{if (___) return; //if input list empty
 else {
     Node<T> *cur = head; Node<T> *next;
     head=0;
     while (cur != 0)

         {___ ;//hold rest of list
           ___ ;//insert odd node
           ___ ;//exit if finished
           ___ ;//skip even node
           ___ ;//move on

         } //end while
     } //end else
 } // end function
```

SOLUTION. For this question, use the following modified Node class definition:

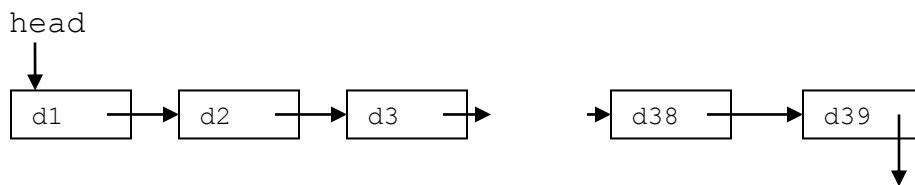
```
template <T>
```

```
class Node {public: T data; Node<T> *link; Node (T d, Node<T> p);}
```

a) (5 pts.) Complete the InsertFront () function that will insert the node pointed by p to the front of the list whose first node is pointed by head.

```
template <class T>
void InsertFront (Node<T> *&head, Node<T> *p)
{if (p==0) return;
 else
    {p -> link = head ;      //link new node to list
      head = p ;}           //link head to new node
}
```

b) (20 pts.) Now, complete the function Rev_odd () using InsertFront (). Rev_odd will return with head pointing to a new list whose items consist of only the odd items of the input list, in reverse order. That is, for the following list:



The call Rev_odd(head) ; returns with:



```
template <class T>
void Rev_odd (Node<T> *&head)           //pointer to list
{if (head == 0) return;                 //if input list empty
 else {   Node<T> *cur = head; Node<T> *next;
          head=0;
          while (cur != 0)

              {next = cur -> link;      //hold rest of list

                InsertFront (head, cur); //insert odd node

                if (next = 0) break;     //exit if finished

                next = next -> link;     //skip even node

                cur = next;              //move on

              } //end while
          } //end else
} // end function
```

Question 3) (18 pts)

You are given a modified version of the Node Class covered in the lectures as follows.

```
template <class T>
class Node
{
public:
    T data;
    Node<T> *next;
    Node (const T& item, Node<T>* ptrnext = NULL);
};
```

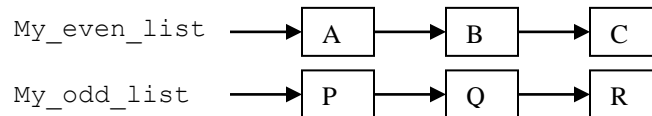
Write a global function

```
Template <class T>
```

```
void Merge(Node<T>*& even_list, Node<T>*& odd_list)
```

which takes two linked lists `even_list` and `odd_list` of the **same size** and merges `odd_list` into `even_list`. As the result of this operation, `even_list` is changed into a new list whose nodes at the odd positions are the nodes of the `odd_list` and nodes at the even positions are the nodes of the original `even_list`. See the figure for an example:

Before Merge (Node<T>*& My_even_list, Node<T>*& My_odd_list)



After Merge (Node<T>*& My_even_list, Node<T>*& My_odd_list)



Fill in the boxes and follow the program comments.

```
template <class T>
void Merge(Node<T>*& even_list, Node<T>*& odd_list)
{
    Node<T>* curr_even= even_list;
    Node<T>* curr_odd= odd_list;
    Node<T>* next_even = curr_even->next();
    Node<T>* next_odd = curr_odd->next();
    if(next_even==NULL) //the lists have one node each
```

```
    curr_even->next=curr_odd;
```

```
    else{
        while(next_even!=NULL) //process all of the nodes in the lists
```

```
        {
            curr_even->next= curr_odd;
            curr_even= next_even;
            next_even= next_even->next;

            curr_odd->next= curr_even;
            curr_odd= next_odd;
            next_odd= next_odd->next;
        }
```

```
    }
    odd_list=NULL;
}
```

Question 4) (25 pts)

Given the node class declaration in the lectures below:

```
template <class T>
class Node
{
private:
Node <T> *next; // next part is a pointer to nodes of this type
public :
T data; // data part is public
// constructor
Node (const T &item, Node<T>* ptrNext=0);

//Insert the node pointed by p after the current node
void InsertAfter(Node<T> *p);
//Delete the node after the current node
Node <T> *DeleteAfter(void);

//get address of next node
Node<T> *NextNode(void) const;
}
```

Part a) 5 points Implement the functions

```
void Node::InsertAfter(Node<T> *p)
{
```

```
    p->next=next;
    next=p;
```

```
    }
Node <T> * Node::DeleteAfter(void) /*delete node following the current node. Do
not dellocate the memory of the deleted node, just return its address */
{
```

```
    //save address of node to be deleted
Node <T> *tempPtr=next;
//if no successor, return NULL
if (next==NULL)
return NULL;
//delete next node by copying its nextptr to the
//nextptr of current node
next=tempPtr->next;
//return pointer to deleted node
return tempPtr;
```

```
}
```

Part b) You are given a linked list that consists of integers and the function

void Arrange2 (Node <T>* &head) . This function rearranges the nodes of the linked list such that the nodes with even integers are at the beginning of the list, followed by the nodes with odd integers.

The order of occurrence among the group of nodes that give the same remainder should be maintained as in the original list.

Example:

Original linked list data: 12, 43, 36, 3, 90,14,2, 67

After calling Arrange2:

12, 36, 90, 14, 2, 43, 3, 67

Implement Arrange2 by filling in the blanks and boxes of the function below according to the given comments.

Hint: mod operation in C++

16 mod 2 : 16%2

template <class T>

void Arrange2 (Node <T>* &head)

{

//declare the variables required to move through the list and perform

//the rearrangement

Node<T> *currPtr=head, *prevPtr=head;

Node<T> *oddhead=NULL;

Node<T> *oddcptr=NULL;

//return if list is empty

if

(currPtr==NULL)

return;

//divide the list into 2, keep even items in the original list, move //odd items to another linked list

while

(currPtr !=NULL)

{

if (currPtr->data%2==1)

{

if (oddhead==NULL)

{

oddhead=GetNode (currPtr->data) ;

oddcptr=oddhead;

}

else

{

oddcptr->InsertAfter (GetNode (currPtr->data)) ;

oddcptr=oddcptr->NextNode () ;

}

if (currPtr==head)

head=head->NextNode () ;

else

prevPtr->DeleteAfter () ;

}

prevPtr=currPtr;

currPtr=currPtr->NextNode () ;

}

//Finalize the function to get the rearranged list

oddcptr=oddhead;

while (oddcptr !=NULL)

{

prevPtr->InsertAfter (GetNode (oddcptr->data)) ;

prevPtr=prevPtr->NextNode () ;

oddcptr=oddcptr->NextNode () ;

}

}

Examples on Linked List:

Q1)

You are given a Linked List that contains items of type T. Nodes are ordered such that stored items are ordered in increasing order.

You are given an item, you create a new node with this item and insert the new node in the correct position such that the list maintains its order.

What is the complexity of this operation in terms of N. (N is the # of items in the list)

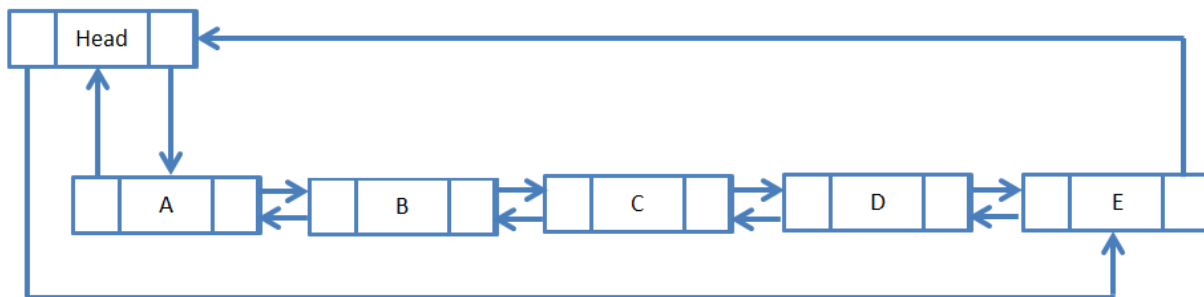
Q2)

```
template <class T>
class nodedouble
{
    private:
        nodedouble<T> *prev;
        nodedouble<T> *next;
    public:
        nodedouble(const T &item, nodedouble<T> *ptrPrev = NULL,
                    nodedouble<T> *ptrNext = NULL);
        void InsertAfter(nodedouble<T> *p);
        nodedouble<T> * DeleteAfter(void);
        nodedouble<T> * NextNode() const;
}
```

(a) Implement “*InsertAfter*” which inserts the node pointed by *p* after the node on which it is applied.

(b) Write a template function “*CreateNode*” that dynamically creates a node of *nodedouble* class and initialize it.

(c)



Write a piece of code that uses functions implemented in (a) and (b) in order to get the list structure given above.

(d) For the Linked List above, draw the resulting linked list of the following code.

```
nodedouble<char> head, *p;  
p = head.next;  
head.next->next->prev = &head;  
head.next = head.next->next;  
p->prev = NULL;  
p->next = NULL;
```

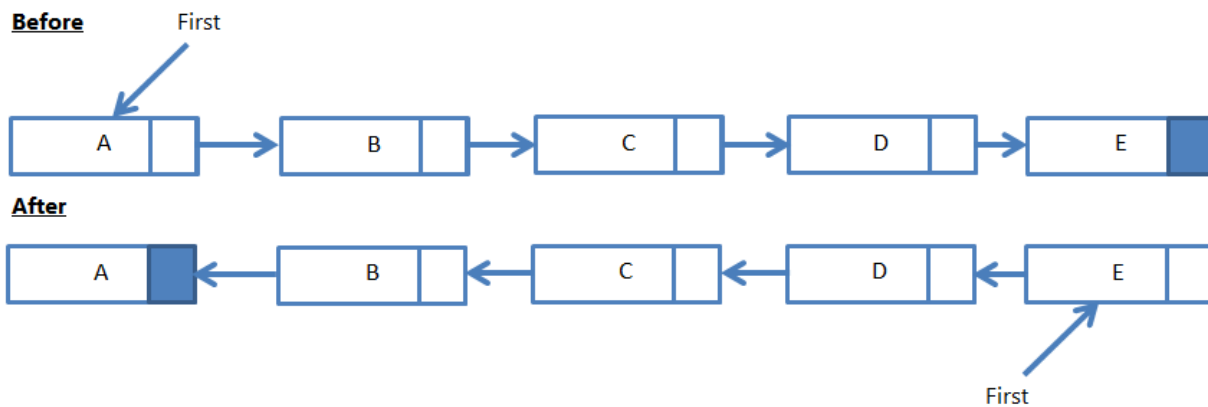
Q3)

Given the NodeClass; (note that Node<T>* next is public), implement a Global function

```
template <class T>
```

```
void Invert(Node<T>* &first);
```

which inverts a given linked list pointed by first by changing only the next pointers.



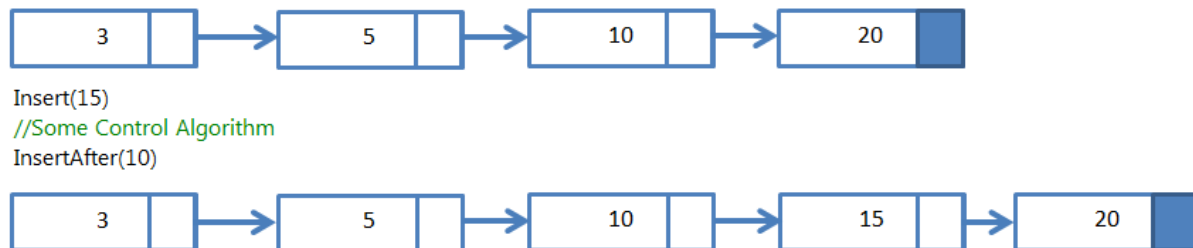
Solutions:

Q1)

The complexity of the operation is $O(N)$ because we can traverse the Linked List only beginning from the head node and progressing node by node.

If we want to insert something in the list, only insertion method for the node is InsertAfter method, as it is described in lecture notes.

i.e.



Q2)

(a) The solution is:

```
void nodedouble<T>::InsertAfter(nodedouble<T> *p)
{
    p->next = next;
    next = p;
    p->next->prev = p;
    p->previous = this;
}
```

OR

```
void nodedouble<T>::InsertAfter(nodedouble<T> *p)
{
    p->next = next;
    next->prev = p;
    p->prev = this;
    next = p;
}
```

(b) The solution is:

```
nodedouble<T>* CreateNode(const T &item, nodedouble<T> *nextPtr = NULL,
                          nodedouble<T> *prevPtr = NULL)
{
    nodedouble<T> *newNode;
    newNode = new nodedouble<T>(item, nextPtr, prevPtr);
    if(newNode==NULL)
    {
        cerr<<"No Memory";
        exit(1);
    }
    return newNode;
}
```

(c) The solution is:

```
nodedouble<char> *head;
head = CreateNode(' ',head,head);
a = CreateNode('A',head,head);
head->next = a;
head->prev = a;
b = CreateNode('B');
a->InsertAfter(b);
c = CreateNode('C');
b->InsertAfter(c);
d = CreateNode('D');
c->InsertAfter(d);
e = CreateNode('E');
d->InsertAfter(e);
e->next = head;
head->prev = e;
```

If this is not a public function, no direct access to *prev* and *next*.
Then;

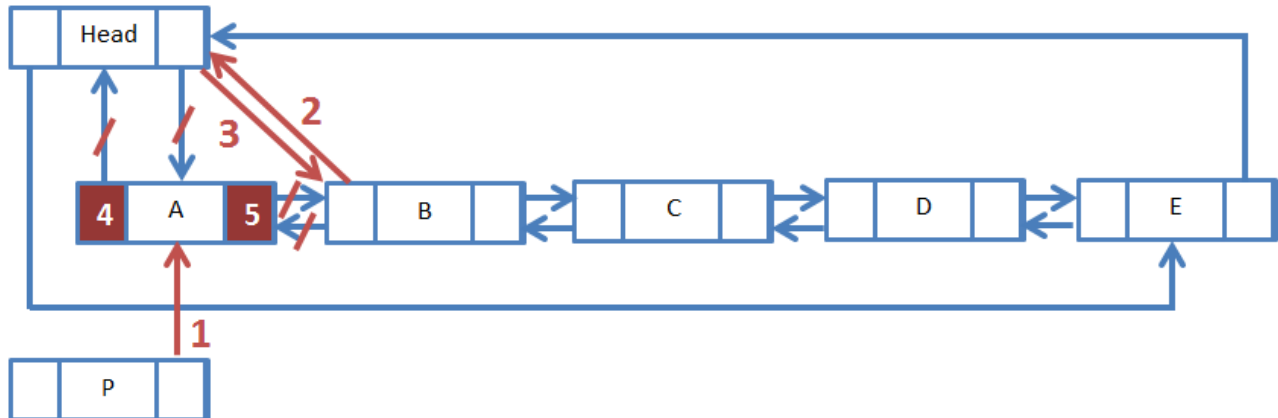
```
a = CreateNode('A');
b = CreateNode('B');
a->InsertAfter(b);
c = CreateNode('C');
b->InsertAfter(c);
d = CreateNode('D');
c->InsertAfter(d);
```

```

e = CreateNode('E');
d->InsertAfter(e);
head = CreateNode(' ');
head->InsertAfter(a);
e->InsertAfter(head);

```

(d) By doing so, we separated Node A from the rest of the linked list.



```

nodedouble<char> *p;
p = head->next; .....1
head->next->next->prev = &head; .....2
head->next = head->next->next; .....3
p->prev = NULL; .....4
p->next = NULL; .....5

```

Q3)

```

void Invert(Node<T>* &first)
{
    Node<T> *temp1, *temp2, *temp3;           //create temporary pointers
    temp1 = first;
    temp2 = NULL;
    temp3 = NULL;
    while(temp1->next != NULL)
    {
        temp3 = temp2;
        temp2 = temp1;
        temp1 = temp1->next;
        temp2->next = temp3;
    }
    temp1->next = temp2;
    first = temp1;
}

```

TREES

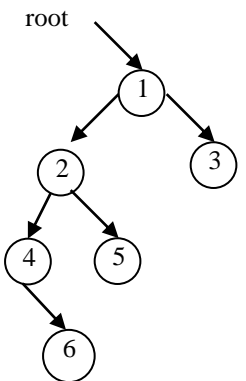
Question 1)

```
template <class T>
class TreeNode
{private:
    TreeNode<T> *left;
    TreeNode<T> *right;
public:
    T data;
    //constructor
    TreeNode(const T &item, TreeNode<T> *lptr=NULL, TreeNode<T>
        *rptr=NULL);

    //access methods for the pointer fields
    TreeNode<T>* Left(void) const;
    TreeNode<T>* Right(void) const;
};

int treeFunc1( TreeNode<int> *t ) {
    if ( t == NULL )
        return 0;
    else {
        cout<< "(";
        int count = 1;
        count = count + treeFunc1(t->Left());
        count = count + treeFunc1(t->Right());
        cout <<"D:"<< t->data <<"C:"<<count<<" ";
        return count;
    }
}

int treeFunc2( TreeNode<int> *t ) {
    if ( t == NULL )
        return 0;
    else {
        int count,countL,countR;
        countL = treeFunc2(t->Left());
        countR = treeFunc2(t->Right());
        count=t->data+countL+countR;
        cout<<"D:"<<t-
>data<<"CL:"<<countL<<"CR:"<<countR<<"C:"<<count<<endl; //endl is new line
        return count;
    }
}
```

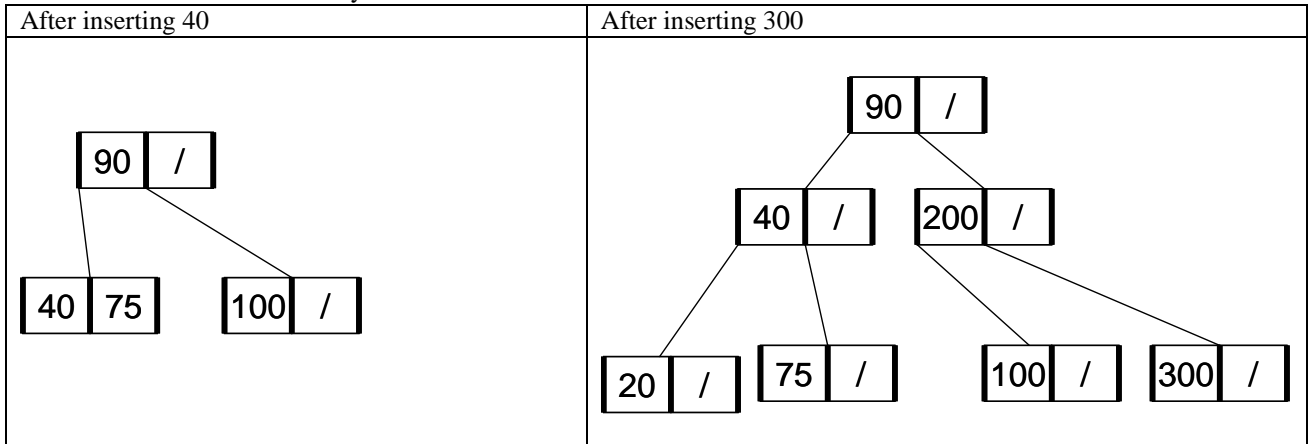
	<p>a) write the output produced when TreeFunc1 (root) is executed:</p> <p>(((D:6:C:1)D:4:C:2) (D:5:C:1)D:2:C:4) (D:3:C:1)D:1:C:6)</p> <p>b) write the output produced when TreeFunc2 (root) is executed:</p> <p>D:6CL:0CR:0C:6 D:4CL:0CR:6C:10 D:5CL:0CR:0C:5 D:2CL:10CR:5C:17 D:3CL:0CR:0C:3 D:1CL:17CR:3C:21</p>
---	--

Question 2) (25 pts)

a) (7 pts.) Insert the following keys in a B-Tree of order 3 in the order they are given.

100,75,90,40,20,200,300

Draw the state of the tree after you insert 40 and 300.



b) (18 pts.) Using the **TreeNode** class in Q3 complete the following **recursive** function (also complete the blank in the function header)

FindKey which searches a given key in a given **binary search tree** that stores integers. The root node of the tree is pointed by MyTree. If key is found, FindKey returns with the pointer to the node that stores the key in argument KeyPointer. If the key is not found, or if MyTree is empty, FindKey returns with NULL value in KeyPointer.

Assume that all items that are stored in the tree exist only once.

```
void FindKey(TreeNode<int> *MyTree, int key, _____KeyPointer)
{
    if (MyTree==NULL)
```

```
    {
        KeyPointer= NULL;
        return;
    }
```

```
    else{
        //stopping condition1: visit the node to check the key match
```

```
    If (MyTree ->data==key)
    {
        KeyPointer= MyTree;
        return;
    }
```

```
    else{
        //If there is no match
        // stopping condition2: if you reach a leaf node, stop the search
        unsuccessfully
```

```
        if(MyTree ->Left()==NULL&& MyTree ->Right()==NULL)
            { KeyPointer= NULL;
              return;}
    }
```

```
    else
        //recursively descend to the correct direction according to the key
```

```
        { if(MyTree ->data>key)
            FindKey (MyTree ->Left(),key);
          else
            FindKey (MyTree ->Right(),key);
        }
```

```
    }
```

Question 3) (25 pts)

a) Given the TreeNode class covered in lectures:

```
template <class T>
class TreeNode
{private:
    TreeNode<T> *left;
    TreeNode<T> *right;
public:
    T data;
    TreeNode(const T &item, TreeNode<T> *lptr=NULL, TreeNode<T>
    *rptr=NULL);
    TreeNode<T>* Left(void) const;
    TreeNode<T>* Right(void) const;
};
```

Write a **recursive** function

```
template <class T>
void int BSTSearch(TreeNode<T> *t, _____level, T key)
```

which takes a given binary search tree and a key. If the key exists in the tree, the function returns the level of the node that the key is found. If the key does not exist, the function returns -1.

Solution:

```
template <class T>
void BSTSearch(TreeNode<T> *t, int& level, T key)
{
    if (t!=NULL)
    {
        if(t->data==key)
            return;
        else
        {
            if(t->Left()==NULL&& t->Right()==NULL)
            {
                level= -1;
                return;
            }
            else if(t->data>key)
            {
                level++;
                BSTSearch(t->Left(),level, key);
            }
            else
            {
                level++;
                BSTSearch(t->Right(),level, key);
            }
        }
    }
}
```

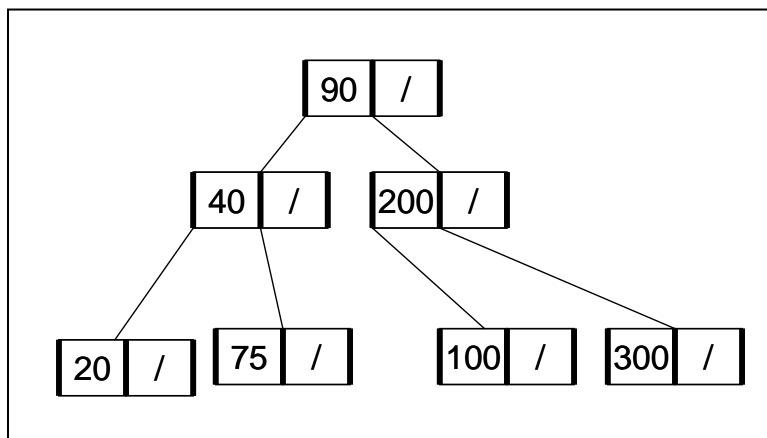
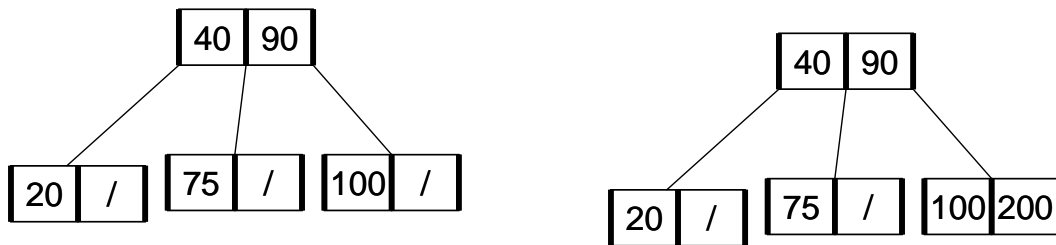
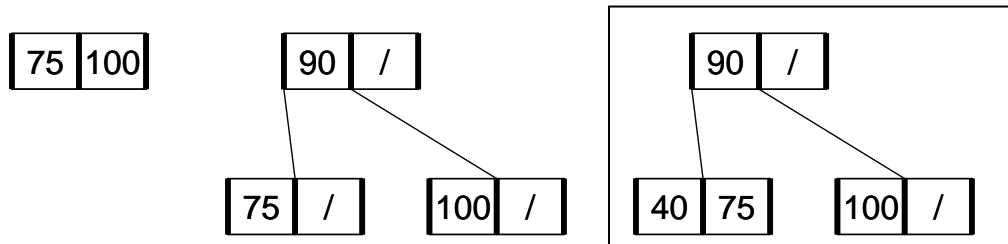
Question 4) (25 pts)

a) Insert the following keys in a B-Tree of order 3 in the order they are given.

100,75,90,40,20,200,300

Draw the state of the tree after you insert 40 and 300.

Solution:



Question 5)

Use the following `TreeNode` class covered in lectures

```
template <class T>
class TreeNode
{private:
    TreeNode<T> *left;
    TreeNode<T> *right;
public:
    T data;
    TreeNode(const T &item, TreeNode<T> *lptr=NULL, TreeNode<T>
    *rptr=NULL);
    TreeNode<T>* Left(void) const;
    TreeNode<T>* Right(void) const;
};
```

a) Write a **recursive** function

```
int TSum (TreeNode<int> *t)
```

which takes a given binary tree which stores positive integers (no 0 and no negative integers) and returns the sum of all stored integers in that tree. If the tree is empty the function returns a 0.

b) Complete the following **recursive** function

```
void TSumLess(TreeNode<int> *MyTree, int key, int& score)
{
    score++;
    ...
}
```

which searches a given key in a given binary search tree that stores positive integers as defined above. The root node of the tree is pointed by `MyTree`. If the key exists in the tree, the function prints the sum of all items that are smaller than or equal to the key on the screen. If the key does not exist, the function does not print anything. Assume that all integers stored in the tree are distinct.

Requirement: After the following function call `myscore` has the **minimum value possible** for the given pointer to `ThisTree` and given key `thiskey`.

```
int myscore=0;
TSumLess(ThisTree, thiskey, myscore);
```

Hint: You might use `TSum` in `TSumLess` (Even you could not solve part a)

SOLUTION

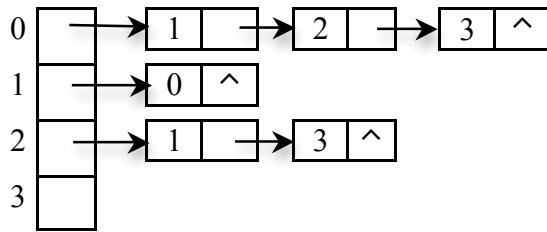
a)

```
template <class T>
int TSum(TreeNode<T> *t)
{
    int sumLeft, sumRight, sum;
    if (t==NULL)
        sum=0; // if empty, sum=0
    else
    {
        sumLeft=TSum(t->Left());
        sumRight=TSum(t->Right());
        sum=t->data+sumLeft+sumRight;
    }
    return sum;
}
```

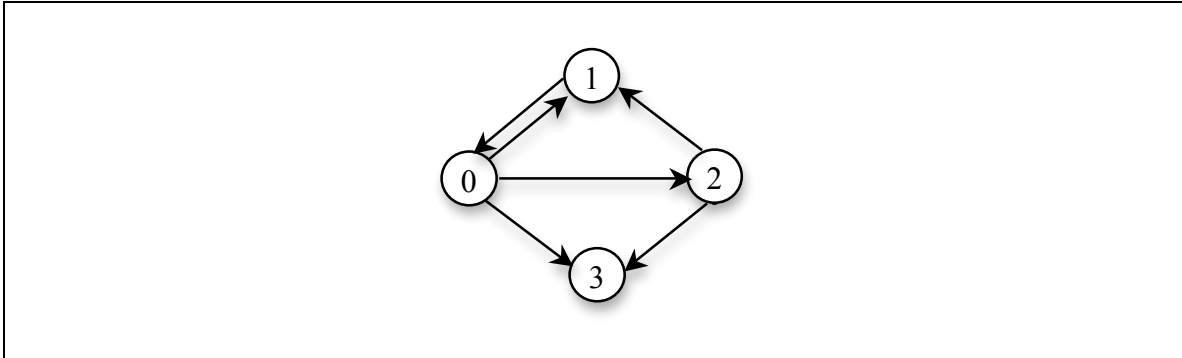
b)

```
void TSumLess(TreeNode<int> *MyTree, int key, int& score)
{
    score++;
    if (MyTree!=NULL)
    {
        if(MyTree ->data==key)
        {
            cout<<TSum(t);
            return;
        }
        else
        {
            if(MyTree ->Left()==NULL&& MyTree ->Right()==NULL)
                return;
            else if(MyTree ->data>key)
                TSumLess(MyTree ->Left(),key, score);
            else
                TSumLess(MyTree ->Right(),key, score);
        }
    }
}
```

Question 8) (6 points) Here is an adjacency list representation of a directed graph:



(a) Draw a picture of the directed graph that has the above adjacency list representation.



(b) Another way to represent a graph is an adjacency matrix. Draw the adjacency matrix for this graph.

	0	1	2	3
0	0	1	1	1
1	1	0	0	0
2	0	1	0	1
3	0	0	0	0

Question 9) (14 points) You are given the Graph class definition as follows

```
class Graph
{public:
    Graph(int n) {}; // Constructor; n: number of vertices
    Graph(const Graph&) {}; // Copy constructor
    ~Graph(void) {}; // Destructor
    int n(); // Returns the number of vertices
    int e(); // Returns the number of edges
    int degree(int v); // Returns the degree of vertex v
    void setEdge(int v1, int v2, int weight); // Set the weight for an edge
    void delEdge(int v1, int v2); // Delete an edge (v1, v2)
    bool isEdge(int v1, int v2); // Determine if the edge (v1, v2) is in
                                // the graph
    int weight(int v1, int v2); // Return weight of the edge (v1, v2):
    int first(int v); // Returns v's first neighbor; Returns
                    // "-1" if there is no neighbor
    int next(int v, int w); // Returns v's next neighbor after w;
                          // Returns "-1" if there is no
                          // more neighbor after w
    ....
    int getMark(int v); // Get mark value (visited=1,
                      // unvisited=0) for vertex v
    void setMark(int v, int val); // Set mark value for vertex v
    void clearMark(void); // Set all mark values as unvisited
};
```

A **tree** is an directed connected graph without cycles. In order to check if a graph G is a tree having node v as root, a modified depth first or breadthfirst traversal algorithm starting with node v may be used. You are requested to complete the following modified breadthfirst graph traversal algorithm to check whether the given undirected graph G corresponds to a tree with root v (returns True) or not (returns False). Assume that G is not empty.

```
int isTree_BreadthFirst(Graph G, int v)
{ // Traverses graph G beginning at vertex v iteratively by using breath first strategy
    const int visited=1, unvisited=0;
    const int True=1, False=0;
    Queue q;
    q.insert(v); // add v to the queue
    G.setMark(v, visited) // Mark v as visited;
    int nodes_visited=1; // number of nodes visited initialised to 1
    while (!q.isEmpty())
    {
        w=q.delete()
        for (u=G.first(w); u!= -1; u=G.next(w, u)) // for each vertex u adjacent to w
        {
            if ( ) // if cycle detected
```

```

        return False;

    else // no cycle detected yet
    {

        q.insert(u);
    } //end of else
} //end of for
} // end of while; traversal completed, therefore no cycle occurred

if ( ( ) ) //if connected

    return True
else
    return False ;

} // end of function

```

Solution:

```

int isTree_BreathFirst(Graph G, int v)
{ // Traverses graph G beginning at vertex v iteratively by using breath first strategy
    const int visited=1, unvisited=0;
    const int True=1, False=0;
    Queue q;
    q.insert(v); // add v to the queue
    G.setMark(v, visited) //Mark v as visited;
    int nodes_visited=1; // number of nodes visited initialised to 1
    while (!q.isEmpty())
    {
        w=q.delete()
        for (u=G.first(w) ; u!= -1 ; u=G.next(w, u) ) // for each
vertex u adjacent to w
        {
            if ( G.getMark(u) == visited ) // if cycle detected

                return False;

            else // no cycle detected yet
            {

                G.setMark(u,visited);
                nodes_visited ++;
            }
        }
    }
}

```

```

        q.insert(u);
    } //end of else
} //end of for
} // end of while; traversal completed, therefore no cycle occurred

if ( nodes_visited == G.n() ) //if connected

    return True
else
    return False ;

} // end of function

```

Question 1) Consider the following partial C++ program:

```
int count(char* &b)
{
    char A[5]={ '1', '2', '3', '4', '5'}, *c, t;
    int X, *p;
    c = A;
    t = *c; *c = *b; *b = t;
    b = c;
    p = &X;
    for (int i=0; i<=4; i++)
        if (A[i] = *b) {X = i; i = 4};
    return *p;
};

void main (void)
{
    char X[3]={ 'M', 'T', 'A'}, *y;
    y = X + 2;
    cout << count(y);
    ...
}
```

Show the contents of all local and global data items. What will be printed out?

Initially in Main:

X: ['M', 'T', 'A']

y points to X[2], passed to count(y).

Initially in Count:

B is a pointer to char, passed by reference,
so it refers to y in Main and points to X[2]

A: ['1', '2', '3', '4', '5']

c points to A[0], t is a char variable

X is a local integer, p is a local ptr-to-integer

t gets value '1', A[0] gets 'A' when "to the location pointed by c, carry the contents of location pointed by b which refers to y" is executed...

X[2] gets value '1' when "to the location pointed by b which refers to y, carry the contents of t" is executed...

y points to A[0] when "copy c into b which refers to y" is executed

p points to the local integer X

upon the first turn of the for loop with i=0, A[0] is found to be equal to itself pointed by y, so the local integer X takes the value of i which is 0, and i is made 4. before the next turn, i is incremented, making it 5, so the loop is not executed any more.

The value of integer X pointed by local p, is returned from count.

So: Output: 0

At the point of output:

X: ['M', 'T', '1']

Y points to A[0] in the local area of count, no longer active.

In the local area of count:

A: ['A', '2', '3', '4', '5'] c points to A[0], t is a char and contains '1'

X is a local integer, contains 0, p is a local ptr-to-int and points to the local integer X

Question 2) Consider each of the following program structures. Determine their $O()$, $\Omega()$ and $\theta()$ time complexities, explaining your reasoning briefly. For all cases, the following time complexities are known:

Func1(n)= $\theta(n)$; Func2(n)= $\theta(2^n)$; Func3(n)= $\theta(\log n)$;
Func4(n)= $O(n)$; Func4(n)= $\Omega(\log n)$.

(a) for (int k=0; k < n; k++)
 if (k%2==0) Func1(n);
 else Func2(n);

$O(2^n)$	$\Omega(2^n)$	$\theta(2^n)$
Reason: <i>Func2() is dominant as $n \rightarrow \infty$</i>		

(b) if (x < A) Func1(n);
 else if (x < A + 1000) Func2(n);
 else if (x < A + 5000) Func3(n);
 else Func4(n);

$O(2^n)$	$\Omega(\log n)$	$\theta(\text{not defined})$
Reason: <i>Best case and worst case behaviours depend on x.</i>		

(c) int Func0(int n)
 { if n < 2 return 2;
 if (n%3 == 0) Func4(n);
 else Func3(n);
 Func0(n/3);
 }

$O(n \log n)$	$\Omega(\log^2 n)$	$\theta(\text{not defined})$
Reason: <i>The number of recursive calls is $\theta(\log_3 n)$. In the best case, in all calls, Func3 is called. In the worst case, in all calls, worst case behaviour of Func4 is encountered.</i>		

Question 3)**a. (5 pts)**

<p>Consider the following C++ class declaration:</p> <pre>class Z { private: int z1; int *z2; public: void Z(int n=0) { z1=n; z2 = new int[z1]; for (int i=0; i<z1; i++) *(z2+i) =i*i; }; int Zread(int j) { return *(z2+j); } }</pre>	<p>Draw the constructed data structures after the execution of the following code:</p> <pre>... Z *A; A = new Z(3); Z B(A -> Zread(1)); ...</pre> <p>Your answer:</p>
---	---

b) (10 pts)

<p>Consider the following C++ class declaration:</p> <pre>template <class T> class K { private: T member1; T *member2 public: K (const T &m1, const T &m2) { member1 = m1; member2 = new T(m2); } }</pre>	<p>i. Draw the constructed data structure after the execution of the following code:</p> <pre>.... K <int> A(20,10), B(700,900); B = A;</pre> <p>Your answer:</p>
---	--

ii. Is there a problem in the solution? What would you do to solve this problem?

c) (10 pts) Assume that class DClass is declared and implemented in file “d.h” with all necessary member functions to handle its dynamic data. Considering the code segment given below complete the following table so that for each executed code line write if constructor, copy constructor, destructor or assignment operator is used and write details as given in the first line as an example.

line# Code

1	# include <iostream.h>
2	# include "d.h"
3	template <class T>
4	DClass<T> MyFunc(DClass<T> A, DClass<T> &B, T m1)
5	{ DClass<T> C(m1);
6	DClass<T> D=C ;
7	C=B;
8	return C;
9	};
10	void main()
11	{ DClass<int> E(100);
12	DClass<int> *g ;
13	g=new DClass<int>(200);
14	E=MyFunc(*g,E,300);
15	delete g ;
16	}

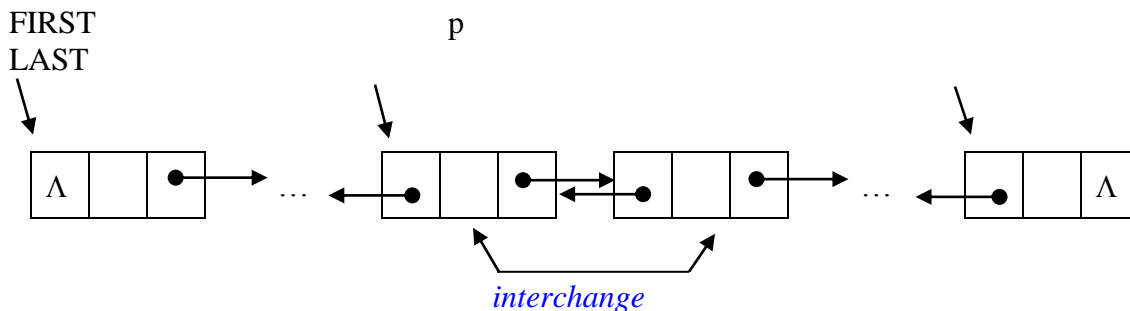
output produced	on object	at line#	by calling DClass member function
A static object E with member variable 100	E	11	Constructor
A dynamic object g with member variable 200	g	13	Constructor
Object A with member variable 200	A	4	Copyconstructor
A static object C with member variable 300	C	5	Constructor
A static object D with member variable 300	D	6	Copyconstructor
Object C with member variable 100	C	7	Assignment operator
Return object with member variable 100	return	8	Copyconstructor
Objects A,C,D are deleted	A,C,D	9	Destructor
Object E with member variable 100	E	14	Assignment operator
Return object is deleted	return	14	Destructor
Dynamic object g is deleted	g	15	Destructor
Static object E is deleted	E	16	Destructor

Question 4) (20 pts.) SOLUTION

Consider the doubly linked list node structure:

prev	data	next
------	------	------

in which **prev** is used to point to the previous node and **next** is used to point to the next node. Assume that FIRST and LAST are pointers to point the first and the last nodes of a doubly linked list, and p is a pointer to a node (not necessarily an intermediate node) in this list. The **prev** field of the node pointed by FIRST and the **next** field of the node pointed by LAST are NULL.



Write a function that will interchange the places of node pointed by p and the next node. *Do not attempt to interchange data fields but change only link fields.* Be careful handling the special cases mentioned below.

```
void Interchange(doublynode<T> &FIRST, doublynode<T> &LAST,
doublynode<T> p)
{if ((p == null) || (p->next == null))
{cout<<"node interchange impossible"; return;}

// adjust prev and next fields as necessary
{ doublynode<T> *q=p->next;

    q->prev=p->prev;
    p->next=q->next;
    if (q->next!=null){q->next->prev=p};
    if (p->prev!=null){p->prev->next=q};
    q->next=p;
    p->prev=q;

// adjust headers as necessary
if(p==FIRST)
// case: p originally pointed to the first node
{
    FIRST=q;
}
if (q==LAST)
// case: the next node was originally the last node
{
    LAST=p;
}
}}
```

Question 5). Consider the following C++ class statement:

```
template <class T>
class X
{public: int n; // number of items in the dynamic array
      p *T;    // dynamic array
/* initializing constructor initializes all entries in the
dynamic array of size nn identically as item */
      X (const int nn=0, const T &item);
      X (const X<T> &xx);
      ~X(void);
      X<T> &operator =(const X<T> &r);}
```

(a) (20 pts.) Implement the initializing and copy constructors, destructor and overloaded assignment operator for this class.

SOLUTION:

```
template <class T> //Initializing constructor
X<T>::X (const int nn=0, const T &item)
{n=nn;
 if(n) {p=new T [n]; for (int i=0; i<n; i++) *(p+i)=item;}};

template <class T> //Copy constructor
X<T>::X (const X<T> &xx)
{n=xx.n;
 if(n) {p=new T [n]; for (int i=0; i<n; i++) *(p+i)=xx.p[i];}};

template <class T> //destructor
X<T>::~~X(void)
{if(n) delete [ ] p;};

template <class T> //Overloaded assignment
X<T> &X<T>:: operator =(const X<T> &r)
{if(n) delete [ ] p;
 n=r.n;
 if(n) {p=new T [n]; for (int i=0; i<n; i++) *(p+i)=r.p[i];}
 return *this;
};
```

Continued at the back of the sheet... →

(b) (5 pts.) Assuming that linked list based implementations of the `Stack<T>` and `Queue<T>` classes are available with the following public methods:

```
void Push(T item); T Pop(void); int StackEmpty(void);  
void QInsert(T item); T QRemove(void); int QueueEmpty(void);  
Stack<T> &operator =(const Stack<T> &r);  
Queue<T> &operator =(const Queue<T> &r);
```

Draw a diagram that shows the data structures created when the following code is executed:

```
void main (void)  
{ Queue<int> A; for (int i=0; i<5; i++) A.QInsert(i);  
  X < Queue <int> > B (2, A);  
  X < int > C (3, 5);  
  Stack <X <int> > D; for (int i=5; i<6; i++) D.Push (C);}
```

SOLUTION:

Question 6) (25 pts) Show all your work. You are given the Node Class and the GetNode function as defined in the lectures. Use **GetNode** if you need to create a new node rather than the constructor.

LStack class uses a linked list for storing the items in the stack. **After a member function that modifies the stack content returns, the number of nodes which consume memory is the same as the number of items stored in the stack.**

```
template <class T>
class Node
{
private:
Node <T> *next;
T data;
Node (const T &item, Node<T>*
ptrNext=0);
void InsertAfter(Node<T> *p);
Node <T> *DeleteAfter(void);
Node<T> *NextNode(void) const;
template <class T>
Node<T> *GetNode(const T& item,
Node<T> *nextPtr=NULL)
```

```
#include "node.h"
template <class T>
class LStack
{
private :
Node<T> *top;
public :
LStack(void); //constructor
to initialize top to NULL for
empty stack
void Push(const T& item);
T Pop(void); };
```

a) Implement the member function Push for the LStack class above in the space provided below.

```
template <class T>
void LStack<T>::Push(const T& item)
{
top=GetNode(item, top);
}
```

b) Implement the member function Pop for the LStack class above by completing the blanks. Pop returns the popped item.

```
template <class T>
T LStack<T>::Pop()
{//Handle the case when the stack is empty
if(top==NULL)
{cerr<<"Stack Empty\n";
exit(1);
}
T popped=top->data;
if(top->NextNode()==NULL) //Handle the case when the stack becomes empty after Pop
{delete top;
top=NULL;}
else
{
//create a new top that is a copy of the item that is below the original top
Node<T>*second=top->NextNode();
top=GetNode(second->data, top);
//Delete the items that are extra copies and get the desired stack content
delete top->DeleteAfter();
delete top->DeleteAfter();
}
return popped;
}
```

LQueue class given below uses a linked list for storing the items in the queue. **After a member function that modifies the queue content returns, the number of nodes which consume memory is the same as the number of items stored in the queue.**

```
#include "node.h"
template <class T>
class LQueue
{
private:
Node<T> *qfirst; //Pointer to the first item in the queue
public:
LQueue(void); //constructor to initialize qfirst to NULL for empty queue
void QInsert(const T& item);
QDelete(void);
void QMovetoFront(int pos); //constructor to initialize qfirst to NULL for empty queue
};
```

- c) Implement the member function QMovetoFront which moves the node at position pos to the front of the Queue. The first item in the Queue is at position 0. Assume pos is always a valid number with respect to the size of the queue and the queue is never empty.

```
template <class T>
void LQueue<T>::QMovetoFront(int pos)
{
//Handle the case when there is only one item in the queue
if(qfirst->NextNode()==NULL)
return;
Node<T> *current=qfirst;
Node<T> *prev=qfirst;
//current and prev pointers move in pair until current is at pos
for(int i=0;i<pos; i++)
{prev=current;
current++;}
//create a new first node with the data content of the current node and delete the extra copy at pos
qfirst=GetNode(current->data, qfirst);
prev->DeleteAfter();
delete current;
}
```

- d) Implement the global template function void QDeletePos (LQueue<T> __Q, int pos) in the space provided below which deletes the item at position pos. Assume pos is always a valid number with respect to the size of the queue and the queue is always non-empty.

```
template <class T>
void QDeletePos(LQueue<T> &Q,int pos)
{
Q.MoveFront(pos);
Q.QDelete();
}
```

Question 7) (a) (8 pts) Assuming that the `TreeNode` structure is defined as in class, complete the missing parts of the following C++ function that searches a given binary search tree for a given key value and returns either a pointer to the node where it is found, or a null pointer value if the key value is not found in the tree:

```
template <class T>
TreeNode<T>* sbt (TreeNode<T>* t, T key)
{ if (t == NULL) return NULL; //nothing to process

    else {if (t -> data == key) return t; //key found

        else {if (t -> data < key) //must search left subtree

            {if (t -> left == NULL) return NULL; //not
there
            else
            return sbt (t -> left, key); //search left

            else {if (t -> right == NULL) return NULL;
            else

                return sbt (t -> right, key); //search right
            } //end key not found at this node
        } //end (sub)tree not null
    } //end sbt
```

(b) (5 pts) What is the $O(\cdot)$ complexity of the function in part (a) in terms of n , the number of items in the tree? Justify your answer with one sentence.

ANSWER: $O(n)$ as each node is visited at most once.

(c) (6 pts) Given the following execution time complexities:

$\text{Func1}(n) = \theta(n)$, $\text{Func2}(n) = \theta(2^n)$, $\text{Func3}(n) = \theta(\log n)$

What are the $\Omega(\cdot)$, $O(\cdot)$ and $\theta(\cdot)$ complexities of the following function in terms of n ? Explain your reasoning briefly:

```
int Funcm (int n)
{ if (n <= 1) return 1;
  if (n <= 100) return Func2(Funcm(n/10));
  if (n <= 1000) return Func3(Funcm(n/10));
  return Func1(Funcm(n/10));
```


SOLUTION:

As $\Omega(\cdot)$, $O(\cdot)$ and $\theta(\cdot)$ are all defined for $n \rightarrow \infty$, only the call to Func1 (Funcm ($n/10$))) has to be considered.

For large n , $\theta(\log_{10} n)$ calls to Funcm are necessary to reduce n to $n \leq 1000$.

Since each call to Func1(n) executes in $\theta(n)$ time, $\text{Funcm}(n) = \theta(n \log n)$

(d) (6 pts) Insert, in the given sequence, the following key values into a B-Tree of order three, based on alphabetical ordering. Draw the resulting B-Tree after the underlined keys are inserted:

ROSE, VIOLET, HYACINTH, BEGONIA, DAFFODIL, ORCHID, NARCISSUS, MAGNOLIA, EDELWEISS, AZALEA, PANSY, MIMOSA.

Question 10) The following numbers are stored in an array are to be sorted:

1	709
2	015
3	702
4	660
5	221
6	162
7	603

a) Show the contents of the array for the first three passes (a “pass” refers to a full turn of the inner loop for a single value of the outer loop index) if Selection Sort is used.

initial	Step1	Step2	Step3
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7

b) Repeat (a) if Bubble Sort is used

initial	Step1	Step2	Step3
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7

c) Repeat (a) if Quicksort is implemented, taking the item with the lowest index as pivot in each pass:

	initial		Step1		Step2		Step3
1	709	1		1		1	
2	015	2		2		2	
3	702	3		3		3	
4	660	4		4		4	
5	221	5		5		5	
6	162	6		6		6	
7	603	7		7		7	

Question 11) (10 pts)

```

void GnomeSort(int A[], int N)
{
    int pos = 1;
    int pass = 0;
    while (pos < N)
        if (A[pos] >= A[pos-1])
            pos = pos + 1;
        else {
            swap(A[pos], A[pos-1]);
            if (pos > 1)
                pos = pos - 1;
            else
                pos = pos + 1;
            pass=pass+1;
            // check point: output pass, A, pos
        } //end if
    } //end while
} //end function

```

a) Given the initial value for array A (note that array size $N=7$), show the content of A and pos as the GnomeSort fuction passes through the check point (show only upto 6 passes).

Initial	pass	1	2	3	4	5	6
A 0		4	4	2	2	2	2
1		6	2	4	4	3	3
2		2	6	6	3	4	4
3		3	3	3	6	6	6
4		1	7	7	7	7	5
5		7	5	5	5	5	7
6		8	8	8	8	8	8
pos	1	2	1	2	2	1	4

b) What are $O(\cdot)$ and $\Omega(\cdot)$ complexities for Gnome Sort

$O(n^2)$, $\Omega(n)$

--

Given the initial values of `first`, `last`, `myArray` and `exchange`, write their values as `ShakerSort` passes through control points `cpt1` and `cpt2`

[illegible]

Question 13) (15 pts) Find out what will be the output if the following code is executed. For each output line, if a part of the line is produced by PrintMembers function, mention the name of the object on which this member function is applied. For example if the constructor is called for object A(1,100), then the output line together with your comment will be as: “Constructor: 1/100 //A”

```
template <class T>
class DC
{
public:
    T member1;
    T *member2;
    DC(const T &m1, const T &m2)
    {
        member1=m1;
        member2=new T(m2);
        cout<<"Constructor: "; PrintMembers( ); };
    DC(const DC<T> &obj)
    {
        member1=obj.member1;
        member2=new T(*obj.member2);
        cout<<"Copy Constructor: "; PrintMembers( ); };
    DC<T>& operator=(const DC<T> rhs)
    {
        member1=rhs.member1;
        *member2=*rhs.member2;
        cout<<"Assignment: "; PrintMembers( );
        return *this;};

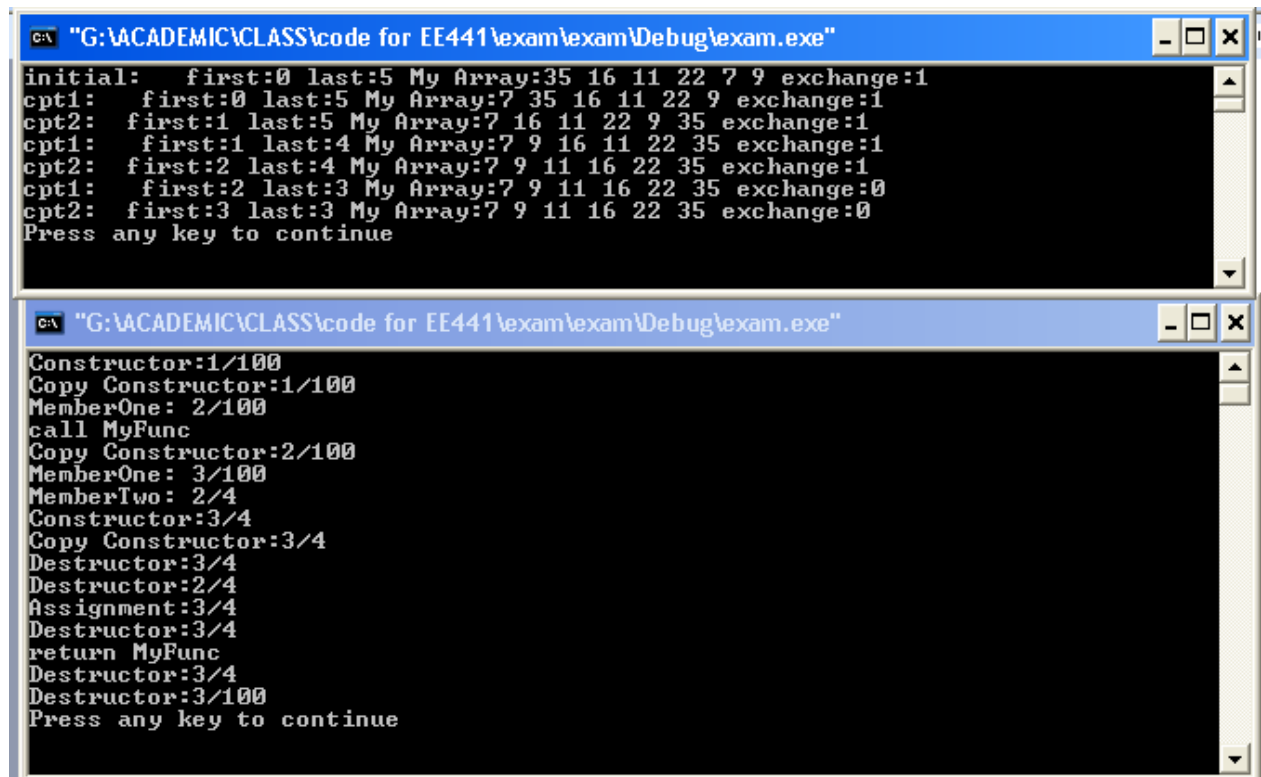
    void MemberOne(T m1)
    {
        member1=m1;
        cout <<"MemberOne: ";PrintMembers( );};
    void MemberTwo(T m2)
    {
        *member2=m2;
        cout <<"MemberTwo: "; PrintMembers( );};
    void PrintMembers (void)
    {
        cout<<member1<<"/"<<*member2<<endl; };
    ~DC(void)
    {
        cout<<"Destructor: "; PrintMembers();
        delete member2;};
};

template <class T>
DC<T> MyFunc(DC<T>& X, DC<T> Y)
{
    X.MemberOne(3);
    Y.MemberTwo(4);
    DC<T> Z(X.member1,*Y.member2);
    return Z;
};

void main( )
{
    DC<int> A(1,100), B=A;
    B.MemberOne(2);
    cout << "call MyFunc"<<endl;
    B=MyFunc(A,B);
    cout << "return MyFunc"<<endl;
}
```

SOLUTION:
 Constructor: 1/100 // A
 CopyConstructor: 1/100 // B
 MemberOne: 2/100 // B
 Call MyFunc
 CopyConstructor:2/100 //Y
 MemberOne: 3/100 //X
 MemberTwo: 2/4 //Y
 Constructor:3/4 //Z
 CopyConstructor:3/4 //return obj
 Destructor: 3/4 //Z
 Destructor:2/4 //Y
 Assignment:3/4 //B
 Destructor:3/4 // return obj
 returnMyFunc
 Destructor:3/4 //B
 Destructor:3/100 //A

Output



The image shows two screenshots of a Windows command prompt window. The title bar of the window is blue and contains the text "C:\ "G:\ACADEMIC\CLASS\code for EE441\exam\exam\Debug\exam.exe".

The first screenshot shows the following output:

```
initial:  first:0 last:5 My Array:35 16 11 22 7 9 exchange:1
cpt1:    first:0 last:5 My Array:7 35 16 11 22 9 exchange:1
cpt2:    first:1 last:5 My Array:7 16 11 22 9 35 exchange:1
cpt1:    first:1 last:4 My Array:7 9 16 11 22 35 exchange:1
cpt2:    first:2 last:4 My Array:7 9 11 16 22 35 exchange:1
cpt1:    first:2 last:3 My Array:7 9 11 16 22 35 exchange:0
cpt2:    first:3 last:3 My Array:7 9 11 16 22 35 exchange:0
Press any key to continue
```

The second screenshot shows the following output:

```
Constructor:1/100
Copy Constructor:1/100
MemberOne: 2/100
call MyFunc
Copy Constructor:2/100
MemberOne: 3/100
MemberTwo: 2/4
Constructor:3/4
Copy Constructor:3/4
Destructor:3/4
Destructor:2/4
Assignment:3/4
Destructor:3/4
return MyFunc
Destructor:3/4
Destructor:3/100
Press any key to continue
```

Question 14) (15 pts) The following overflow handling method uses a collision resolution function in addition to the hash function. If $\text{hash}(\text{key})$ results in no collision key is stored at $\text{hash}(\text{key})$. If collision occurs, next probes are performed by considering the formula:

$$h_i(\text{key}) = (\text{hash}(\text{key}) + g(i)) \% N$$

where:

hash(key) is the hash function

g(i) is the collision resolution function

i=1, 2 ... is the number of the current attempt (probe) to insert an element

N is the hash table size

a)

Let $N=8$

$\text{hash}(\text{key}) = \text{key} \% N$

$g(i) = i * i$

Insert the following keys into the hash table in the given order:

10, 3, 2, 14, 6

(not possible to insert 6)

	keys
0	
1	
2	10
3	3
4	
5	
6	2
7	14

b) What is the problem with the resulting $h_i(\text{key})$ when the collision resolution function $g(i)$ given in part (a) is used

The resulting $h_i(\text{key})$ function may repeat itself before an empty position is reached, which happens when 6 is to be inserted.

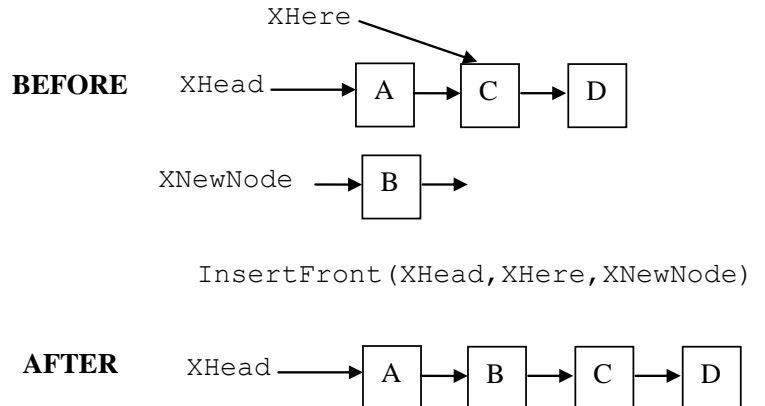
c) propose a $g(i)$ function such that the resulting method is equivalent to linear probing

$g(i) = i$

Question 15)

a) Using the Node class below complete the following global function InsertFront which inserts a given node pointed by NewNode in front of a node in a given Linked List. The Linked List is identified by a pointer to its head that is called Head. The insertion is in front of the node that is denoted by pointer Here. After InsertFront is called, it should be possible to conduct further operations on the Linked List. Assume that the LinkedList is not empty. The call of InsertFront on an example LinkedList pointed by XHead is shown in the figure below.

```
template <class T>
class Node
{
private:
Node <T> *next;
public :
T data;
Node (const T &item, Node<T>* ptrNext=0);
void InsertAfter(Node<T> *p);
Node <T> *DeleteAfter(void);
Node<T> *NextNode(void) const;
}
```



Implement in the given space below. Also complete the blank in the function header.

```
void InsertFront(Node<T>_____Head, Node<T>* NewNode, Node<T>*
Here)
{ Node<T>* current=Head, * prev =Head;
//Traverse the list until Here is found
```

```
while(_____)
{
}
}
```

```
//Special case: Insert in front of the Head Node
```

```
if (Here==Head)
{
}
}
```

```
//No special case, insert simply in front of Here
```

```
else
{
}
}
```

```
}
```


b) UTEM university has 8 students. The student records are stored in a hash table of size 8 (bucket size=1). The student ID expressed in binary form is the key used for hashing. The student IDs are 5 bits. The Student list and their IDs are as follows:

ID (in binary)	Name Last Name
11000	Ellen Ripley
11010	Luke Skywalker
11100	Han Solo
11110	Vincent Vega
10000	Tyler Durden
10010	Lester Burnham
10100	Sarah Connor
10110	Marty McFly

i) What is the key density of this hash table

ii) Find a perfect hash function for this set of student IDs such that no two keys are synonyms. There will be no collision in the resulting hash table. State your hash function and show your resulting key placement in the hash table below:

Hash address	ID (key)
000	
001	
010	
011	
100	
101	
110	
111	

Hash Function:

Solution:

```
void InsertFront(Node<T>* & Head, Node<T>* NewNode, Node<T>* Here)
{ Node<T>* current=Head, * prev =Head;
```

```
//Traverse the list until Here is found
while(current!=Here)
{
prev=current;
current=current->NextNode();
}
```

```
//Special case: Insert infront of the Head Node
```

```
if(Here==Head)
{
NewNode->InsertAfter(Head);
Head=NewNode;
}
```

```
//No special case, insert simply in front of Here
else
{
prev->InsertAfter(NewNode);
}
}
```

b)

i) What is the key density of this hash table= $8/32=0.25$

Hash address	ID (key)
000	10000
001	10010
010	10100
011	10110
100	11000
101	11010
110	11100
111	11110

Hash Function: Take the middle 3 bits and use for addressing

Question 16. (x pts) Explain your answers with a single verbal sentence or a short proof.

- I. Problem P1 is shown to be NP-complete. Problem P2 can be reduced to P1 using an algorithm A. A is $O(N^2)$

a) Can you find a polynomial time solution for P1?

NO (YES ONLY IF $NP=P$)
 $P1 \in NP_Complete$, no known algorithm in P for P1
otherwise it would be $NP=P$

b) Can you find a polynomial certifier for P1?

YES
 $P1 \in NP_Complete \Rightarrow P1 \in NP \Rightarrow \text{Certifier}(P1) \in P$ exist by definition of NP

c) Can you find a polynomial certifier for P2?

$P2 \leq_p P1$ is given \Rightarrow also $P2 \in NP$ (notice that $P \subseteq NP$)
 $\Rightarrow \text{Certifier}(P2) \in P$ exist by definition of NP

d) What are the possible problem complexity classes for P2?

certainly $P2 \in NP$, possibly $P2 \in P$ also

- II. Problem Q1 is known to be NP-complete. Problem Q2 is known to be NP. Problem Q1 can be polynomially reduced to Q2. Problem Q3 is known to be NP. Can you find a polynomial time algorithm to reduce Q3 to Q2?

YES:

$Q1 \in NP_Complete$ and $Q1 \leq_p Q2 \Rightarrow Q2 \in NP_Complete$ by theorem in lecture notes

$Q3 \in NP$, $Q2 \in NP_Complete \Rightarrow Q1 \leq_p Q2$ by definition of NP-completeness

So a polynomial time algorithm to reduce Q3 to Q2 exist.

E2. (x pts) Show all your reasoning and work.

Consider the following function:

```
void myF(int n, int m)
{
    cout<<"n:"<<n<<" m:"<<m<<"\n";
    if (n==1)
        return;
    if (n>m)
        myF(n/2,m);
    else myF(n-1,m);
}
```

a) What is the output of: myF(16, 4);

```
n: 16  m:4
n: 8   m:4
n: 4   m:4
n: 3   m:4
n: 2   m:4
n: 1   m:4
```

b) Assume both m and n, where $m \geq 1$, $n \geq 1$, are always powers of 2, **Derive** the worst case complexity of myF **as a function of n** in $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$. Hint: First derive the complexity in terms of n and m. Then look at the possible relationships between n and m.

Without using hint:

Worst case occur when $n \leq m$, so always else part i.e. myF(n-1,m) is executed, so $O(n)$

Best case occurs when $m=1$, so always if part i.e. myF(n/2,m) is executed, so $\Omega(\log(n))$

If hint is used:

The complexity in terms of m and n is:

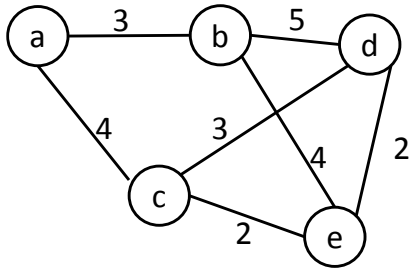
$\theta(\min(m,n) + \max(\log(n/m), 0)) = \theta(m + \max(\log(n) - \log(m), 0))$

Worst case $n \leq m$, so $n/m < 1$ and $\max(\log(n/m), 0) = 0$, $\min(m,n) = m$, therefore $O(n)$

Best case $m=1$, so $n/m = n$ and $\max(\log(n/m), 0) = \log(n)$, $\min(m,n) = 1$, therefore so $\Omega(\log(n))$

Q.5 (20 pts.) Show all your reasoning and work.

Consider the undirected weighted graph below. For the rest of the question, if there is more than one way to take an action follow the alphabetical order.



- a) Show the traversal order if the recursive depth first traversal algorithm $dft(a)$ is called. Indicate the function calls $dft()$ and returns.

- b) Run the iterative depth first algorithm starting from node a. Fill in the following table. Add as many rows as you need.

Node visited	Push	Stack bottom to top	Pop
a	a	a	--

- c) Run the iterative breadth first algorithm starting from node a. Fill in the following table. Add as many rows as you need.

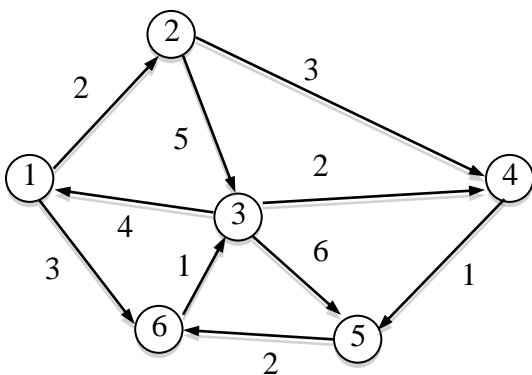
Node visited	Queue (front to back)

- d) Run Dijkstra's shortest path algorithm starting from node a. Fill in the following table. For each iteration indicate the current node, the updated previous node and distance for each node. (NA: not applicable)

INITIAL	Vertex	a	b	c	d	e
	Previous	NA	--	--	--	--
	Distance	0	∞	∞	∞	∞
Current Node:	Vertex	a	b	c	d	e
	Previous	NA				
	Distance	0				
Current Node:	Vertex	a				
	Previous	NA				
	Distance	0				
Current Node:	Vertex	a				
	Previous	NA				
	Distance	0				
Current Node:	Vertex	a				
	Previous	NA				
	Distance	0				
Current Node:	Vertex	a				
	Previous	NA				
	Distance	0				
Current Node:	Vertex	a				
	Previous	NA				
	Distance	0				

Q4. (25 points)

Consider the following directed graph having weighted edges.



a) Represent the graph using adjacency matrix notation

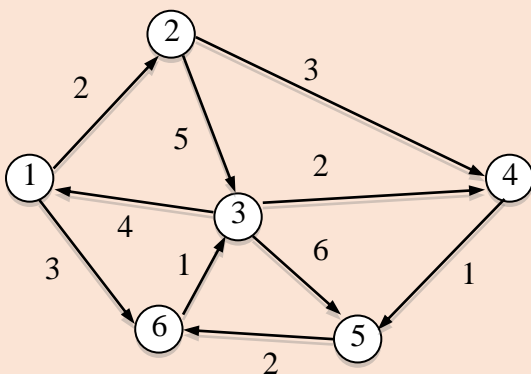
b) Represent the graph using adjacency list notation

c) We want to use Dijkstra's algorithm to determine the shortest path from vertex 1 to each of the other vertices. In the table below, list the vertices in the order that they are visited as the algorithm progresses; also update the entries indicating the current shortest known distance and predecessor vertex on the path from vertex 1 to each vertex. The initial values for the distances are given for you.

Node visited	Current shortest known distance						Predecessor vertex on the path					
	1	2	3	4	5	6	1	2	3	4	5	6
	0	∞	∞	∞	∞	∞	-	-	-	-	-	-
1	0	2										

Q4. SOLUTION

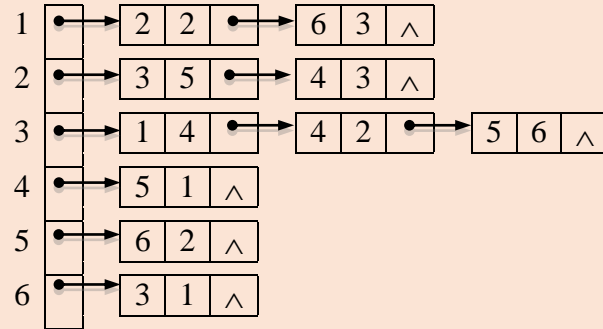
Consider the following directed graph having weighted edges.



a) Represent the graph using adjacency matrix notation

	1	2	3	4	5	6
1	0	2	0	0	0	3
2	0	0	5	3	0	0
3	4	0	0	2	6	0
4	0	0	0	0	1	0
5	0	0	0	0	0	2
6	0	0	1	0	0	0

b) Represent the graph using Adjacency list notation



c) We want to use Dijkstra's algorithm to determine the shortest path from vertex 1 to each of the other vertices. Update the entries in the following table to indicate the current shortest known distance and predecessor vertex on the path from vertex 1 to each vertex as the algorithm progresses. The initial values for the distances are given for you. Below the table, list the vertices in the order that they are visited as the algorithm is executed

Node visited	Current shortest known distance						Predecessor on the path					
	1	2	3	4	5	6	1	2	3	4	5	6
	0	∞	∞	∞	∞	∞	-	-	-	-	-	-
1	0	2	∞	∞	∞	3	-	1	-	-	-	1
2	0	2	7	5	∞	3	-	1	2	2	-	1
6	0	2	4	5	∞	3	-	1	6	2	-	1
3	0	2	4	5	10	3	-	1	6	2	3	1
4	0	2	4	5	6	3	-	1	6	2	4	1
5	0	2	4	5	6	3	-	1	6	2	4	1

Consider the following recursive function:

```

int RecF( int x, int n)
{ int temp=0;
  if (n<=1)
  {temp=0; cout<<"/"<<n<<":"<<temp; return temp;}
  else
  if (n%2==0) // n is even
  {temp=x*x + RecF(x, n/2);
   cout<<"/"<<n<<":"<<temp;
   return temp; }
  else // n is odd
  {temp= x + RecF(x, n-2);
   cout<<"/"<<n<<":"<<temp;
   return temp; }
}
  
```


}

a) What will be the output produced if RecF(3,7) is executed ?

/1:0/3:3/5:6/7:9Press any key to continue

b) Repeat (a) for RecF(3,8).

/1:0/2:9/4:18/8:27Press any key to continue . . .

c) Write the recurrence relation for RecF in terms of n

$$T(n) = \begin{cases} \theta(1) & n < 1 \\ \theta(1) + T(n/2) & n \text{ is even, } n \geq 1 \\ \theta(1) + T(n-2) & n \text{ is odd, } n \geq 1 \end{cases}$$

d) What is the best case performance $\Omega(\cdot)$

$\Omega(\log n)$

e) What is the worst case performance $O(\cdot)$

$O(n)$

Q.6 (15pts.) Show all your reasoning and work.

a) Complete the following statements

i) A problem A is P if _____

ii) A problem A is NP if _____

iii) Given two problems A, B, we say that A is polynomially **reducible** to B ($A \leq_p B$) if:

- _____

- _____

iv) A problem B is NP-complete if:

- _____

- _____

v) A problem B is NP-hard if:

b) Problem X has a polynomial certifier C.

i) Is it possible that X is P? Explain.

Yes, solution is certifier itself.

P is subset of NP

ii) Is it possible that X is EXP? Explain.

Yes NP is subset of EXP

iii) Is it possible that X is NP-hard? Explain

No, has a P-certifier

c) Given $B \in \text{NP}$, $C \in \text{NP-Complete}$. It is known that $C \leq_p D$ and there is no polynomial certifier for D.

i) What are the possible classes for D? Explain.

D is NP-hard by definition

ii) Is $B \leq_p D$? Explain.

C is NP-complete, can solve anything in NP so $B \leq_p C$

Given: $C \leq_p D$ so $B \leq_p C \leq_p D$

by transitivity : $B \leq_p D$