Assignment No :- 1

Program Statement:-

Create a class FLOAT that contains one float data member, overload all the four arithmetic operators so that operate on the objects FLOAT.

Program Algorithm:-

Description:

This is a program where we create a user defined data type of name FLOAT, where we have a data part of floating type identified by f. In this program we overload all arithmetic operators to operate the user define data type FLOAT. The data part of the user define data type is in the private section and other functions that operates with the data part are in the public part.

Steps:

Step 1: creating the **FLOAT(a ← 0)** the default argument constructor
[This is a special function because it's call where an object of the FLOAT data type is created . Here we passes a argument a which is assigned to 0 that means if at the time of cresting a object if no values are passes then it assign the 0 value to the argument a.]

Step 1.1: f ← a

Step 2: creating the **set_data()** function
[This function is used to set the data part of the object.]

Step 2.1: print "Enter a Floating Number ="
input f
Step 2.2: STOP

Step 3: creating the **display()** function
[This is a function that display the data part content in the monitor.]

Step 3.1: print "Value =", f
Step 3.2: STOP

Step 4: creating a function for **overload the operator +(&a)**
[This function is overloaded the + operator and takes a reference type argument of FLOAT data type 'a' and it's returns a value of FLOAT data type.]

Step 4.1: temp is a object of FLOAT data type
Step 4.2: temp.f ← f + a.f
Step 4.3: return temp

Step 4.4: STOP

Step 5: creating a function for **overload the operator -(&a)**
[This function is overloaded the - operator and takes a reference type argument of FLOAT data type 'a' and it's returns a value of FLOAT data type.]

Step 5.1: temp is a object of FLOAT data type
Step 5.2: temp.f ← f - a.f
Step 5.3: return temp
Step 5.4: STOP

Step 6: creating a function for **overload the operator *(&a)**
*[This function is overloaded the * operator and takes a reference type argument of FLOAT data type 'a' and it's returns a value of FLOAT data type.]*

Step 6.1: temp is a object of FLOAT data type
Step 6.2: temp.f ← f * a.f
Step 6.3: return temp
Step 6.4: STOP

Step 7: creating a function for **overload the operator /(&a)**
[This function is overloaded the / operator and takes a reference type argument of FLOAT data type 'a' and it's returns a value of FLOAT data type.]

Step 7.1: temp is a object of FLOAT data type
Step 7.2: temp.f ← f / a.f
Step 7.3: return temp
Step 7.4: STOP

Step 8: creating the **MAIN()**

Step 8.1: a,b,c are the objects of FOLAT data type
Step 8.2: a.display()
b.display()
c.display()
Step 8.3: print "For A"
a.set_data()
print "For B"
b.set_data()
Step 8.4: c ← a+b
print "After C = A+B"
print "C ="
c.display()
Step 8.5: c ← a-b
print "After C = A-B"
print "C ="
c.display()
Step 8.6: c ← a*b

print “After C = A*B”
print “C =”
c.display()

Step 8.7: c ← a/b
print “After C = A/B”
print “C =”
c.display()

Step 8.8: STOP

Program Source Code:-

```
// Header File
#include<iostream>

using namespace std;

// Class FLOAT
class FLOAT
{
	private:
		float f;
	public:
		FLOAT(float a=0)		// Default Argument Constructor
		{
			f=a;
		}

		// A member function to set the value into the object
		void set_data()
		{
			cout<<"enter a Floating Number =";
			cin>>f;
		}

		// A member function to display the value of the object
		void display()
		{
			cout<<"Value ="<<f<<endl;
		}

		// A member function to overload the '+' operator
		FLOAT operator + (FLOAT &a)
		{
			FLOAT temp;
			temp.f=f+a.f;
			return temp;
		}

		// A member function to overload the '-' operator
		FLOAT operator - (FLOAT &a)
```

```
		{
			FLOAT temp;
			temp.f=f-a.f;
			return temp;
		}

		// A member function to overload the '*' operator
		FLOAT operator * (FLOAT &a)
		{
			FLOAT temp;
			temp.f=f*a.f;
			return temp;
		}

		// A member function to overload the '/' operator
		FLOAT operator / (FLOAT &a)
		{
			FLOAT temp;
			temp.f=f/a.f;
			return temp;
		}
};


main()
{
	FLOAT a,b,c;
	cout<<"Initial Value of A,B,C :-\n";
	cout<<"A: ";
	a.display();
	cout<<"B: ";
	b.display();
	cout<<"C: ";
	c.display();
	cout<<"\nFor A ";
	a.set_data();
	cout<<"For B ";
	b.set_data();
	c=a+b;
	cout<<"\nAfter C=A+B\n";
	cout<<"C: ";
	c.display();
	c=a-b;
	cout<<"\nAfter C=A-B\n";
	cout<<"C: ";
	c.display();
	c=a*b;
	cout<<"\nAfter C=A*B\n";
	cout<<"C: ";
	c.display();
	c=a/b;
```

```
        cout<<"\nAfter C=A/B\n";
        cout<<"C: ";
        c.display();
}
```

Output:-

SET 1:-

```
Initial Value of A,B,C :-
A: Value =0
B: Value =0
C: Value =0

For A enter a Floating Number =12.5
For B enter a Floating Number =45.6

After C=A+B
C: Value =58.1

After C=A-B
C: Value =-33.1

After C=A*B
C: Value =570

After C=A/B
C: Value =0.274123

--------------------------------
Process exited after 13.89 seconds with return value 0
Press any key to continue . . .
```

SET 2:-

```
Initial Value of A,B,C :-
A: Value =0
B: Value =0
C: Value =0

For A enter a Floating Number =23.45
For B enter a Floating Number =21.35

After C=A+B
C: Value =44.8

After C=A-B
C: Value =2.1

After C=A*B
C: Value =500.658
```

After C=A/B
C: Value =1.09836

Process exited after 20.3 seconds with return value 0
Press any key to continue . . .

PRECAUTIONS AND DISCUSSION:

DISCUSSION:

1. The assignment operator (=) is not overloaded because this operator is already overloaded in C++ library.
2. Operator overloading cannot change the precedence of operators and associatively of operators. To change the order of evaluation, parenthesis should be used.
3. In case of operator overloading, the object on the right hand side of the operator is always assumed as argument by compiler.
4. In this program we could have used a template class to accept all kinds of data instead of just float.

PRECAUTIONS:

1. Each line is ended with ";"(semicolon) to maintain the format rules.
2. The closing parameter of the class is followed by a ";"(Semicolon)
3. Namespace and proper header files are included to ensure proper execution of the library functions.
4. An infinite loop is used for proper execution according to the user.
5. Proper exit conditions are given to avoid incorrect outputs.
6. The array input for binary search must be in sorted order(for this program it has to be in ascending order).
7. If else condition is used for better execution of the program.

Assignment No :- 2

Program Statement:-

Write a program to carry out polynomial addition and multiplication using linked list ,the linked class should be of template type.

Program Algorithm:-

Description:

This is a program that implements polynomial function creation and displaying of polynomial functions. We can add two polynomial and multiply also by the functions. Here we create a user defined data type named polynomial which have a coefficient part identified by c , a exponent part identified by e and a address part that can hold the address of the polynomial data type identified by next. The c, e, next are in the private part and the other functions are in the public section.

Steps:

Step 1: creating the **polynomial(a ← 0,b ← 0)**
[This the constructor it takes two argument a and b which are passes at the time of creating the object, if it not passes then both a, b are assigned to 0]

Step 1.1: c ← a
e ← b
next ← NULL

Step 2: creating the function **create()**
[This function is use to create a polynomial function where we the coefficient and the exponent from the user]

Step 2.1: p and q are two pointers of polynomial data types
a ← 'y'
Step 2.2: Loop continue when a = 'Y' or a = 'y' true
Step 2.2.1: print "Enter the Coefficient = "
read c
print "Enter the exponent ="
read e
Step 2.2.2: create a object of polynomial data type by polynomial(c,e) and return it's address to the p
Step 2.2.3: if next[this] = NULL then
next[this] ← p
else
next[q] ← p
[end of if]

Step 2.2.4: q ← p
print “Do you want to continue (y/n) =”
read a
[end of Loop]
Step 2.3: STOP

Step 3: creating the function **display()**
[This is a function that display the polynomial function into the monitor]

Step 3.1: create a pointer p of polynomial data type
p ← next[this]
Step 3.2: if p = NULL then
print “No Polynomial function exists.”
STOP
[end of if]
Step 3.3: if c[p] > 0 then
print c[p] , “X^”, e[p], “ ”
else
print -1*c[p], “X^”, e[p], “ ”
[end of if]
Step 3.4: Loop continue when p != NULL true
if c[p] > 0 then
print c[p] , “X^”, e[p], “ ”
else
print -1*c[p], “X^”, e[p], “ ”
[end of if]
p ← next[p]
[End of Loop]
Step 3.5: STOP

Step 4: creating the function **add(a,b)**
[This is a function that takes two arguments and add them and set that polynomial function to the calling object]

Step 4.1: p, q, r, t are pointers of polynomial data type
p ← a.next
q ← b.next
Step 4.2: Loop continue when p != NULL and q != NULL true
Step 4.2.1: if e[p] = e[q] then
r ← a address of a new object by polynomial(c[p]+c[q],e[p])
p ← next[p]
q ← next[q]
Step 4.2.2: else
if e[p] > e[q] then
r ← a address of a new object by polynomial(c[p], e[p])
p ← next[p]
else
r ← a address of a new object by polynomial(c[q], e[q])
q ← next[q]

```
                    [end of if]
                [end of if]
Step 4.2.3:     if next[this] = NULL then
                        next[this] ← r
                else
                        next[t] ← r
                [end of if]
Step 4.2.4:     t ← r
            [end of Loop]
Step 4.3:   Loop continue when p!= NULL true
Step 4.3.1:     r ← a address of a new object by polynomial(c[p], e[p])
                p ← next[p]
Step 4.3.2:     if next[this] = NULL then
                        next[this] ← r
                else
                        next[t] ← r
                [end of if]
Step 4.3.3:     t ← r
            [End of Loop]
Step 4.4:   Loop continue when q!= NULL true
Step 4.4.1:     r ← a address of a new object by polynomial(c[q], e[q])
                q ← next[q]
Step 4.4.2:     if next[this] = NULL then
                        next[this] ← r
                else
                        next[t] ← r
                [end of if]
Step 4.4.3:     t ← r
            [End of Loop]
Step 4.4:   STOP


Step 5:     creating the function mul(a,b)
            [This is a function that takes two arguments and multiply them and set that
            polynomial function to the calling object ]

Step 5.1:   q, p, r, t are the pointers of the polynomial data type
            h1, h2 are the objects of polynomial  data type
            p ← a.next
            next[this] ← NULL
Step 5.2:   Loop continue when p != NULL true
Step 5.2.1:     h1.next ← NULL
                q ← b.next
Step 5.2.2:     Loop continue when q!= NULL true
                        r ← address of a new object by polynomial(c[p]*c[q],e[p]*e[q])
                        if h1.next = NULL then
                                h1.next ← r
                        else
                                next[t] ← r

                        [end of if]
```

t ← r
q ← next[q]
[end of Loop]

Step 5.2.3: h2.next ← next[this]
next[this] ← NULL
this →add(h1,h2)
p ← next[p]
[end of Loop]

Step 5.3: STOP

Step 6: Creating the **Main()** function

Step 6.1: h1, h2, h3 are the object of polynomial data type
Step 6.2: h1.create()
h1.display()
Step 6.3: h2.create()
h2.display()
Step 6.4: h3.add(h1,h2)
h3.display()
Step 6.5: h3.mul(h1,h2)
h3.display()

Program Source Code:-

```
// Header File
#include<iostream>
#include<cstdlib>

using namespace std;

template<class T>
class polynomial
{
	private:
		T c;
		int e;
		polynomial *next;
	public:
		polynomial(T a=0,int b=0)
		{
			c=a;
			e=b;
			next=NULL;
		}

		void create()
		{
			polynomial *p,*q;
			T c;
```

```
        int e;
        char a='y';
        while(a=='Y'||a=='y')
        {
                cout<<"\nEnter the cofficent =";
                cin>>c;
                cout<<"Enter the exponent =";
                cin>>e;
                p=new polynomial(c,e);
                if(this->next==NULL)
                 this->next=p;
                else
                 q->next=p;
                q=p;
                cout<<"Do you want to continue (y/n) =";
                cin>>a;
        }
}

void display()
{
        polynomial *p;
        p=this->next;
        if(p==NULL)
        {
                cout<<"No Polynomial Function Exsits.";
                return;
        }
        if(p->c>0)
         cout<<p->c<<"X^"<<p->e<<" ";
        else
         cout<<"- "<<-1*p->c<<"X^"<<p->e<<" ";
        p=p->next;
        while(p!=NULL)
        {
                if(p->c>0)
                 cout<<"+ "<<p->c<<"X^"<<p->e<<" ";
                else
                 cout<<"- "<<-1*p->c<<"X^"<<p->e<<" ";
                p=p->next;
        }
}

void add(polynomial a,polynomial b)
{
        polynomial *p,*q,*r,*t;
        p=a.next;
        q=b.next;
        while(p!=NULL and q!=NULL)
        {
                if(p->e==q->e)
```

```
                {
                        r=new polynomial(p->c+q->c,p->e);
                        p=p->next;
                        q=q->next;
                }
                else
                 if(p->e>q->e)
                 {
                        r=new polynomial(p->c,p->e);
                        p=p->next;
                 }
                 else
                 {
                        r=new polynomial(q->c,q->e);
                        q=q->next;
                 }
                if(this->next==NULL)
                 this->next=r;
                else
                 t->next=r;
                t=r;
        }
        while(p!=NULL)
        {
                r=new polynomial(p->c,p->e);
                p=p->next;
                if(this->next==NULL)
                 this->next=r;
                else
                 t->next=r;
                t=r;
        }
        while(q!=NULL)
        {
                r=new polynomial(q->c,q->e);
                q=q->next;
                if(this->next==NULL)
                 this->next=r;
                else
                 t->next=r;
                t=r;
        }
}

void mul(polynomial a,polynomial b)
{
        polynomial h1,h2,*p,*q,*r,*t;
        p=a.next;
        this->next=NULL;
        while(p!=NULL)
        {
```

```
                                h1.next=NULL;
                                q=b.next;
                                while(q!=NULL)
                                {
                                        r=new polynomial(p->c*q->c,p->e*q->e);
                                        if(h1.next==NULL)
                                         h1.next=r;
                                        else
                                         t->next=r;
                                        t=r;
                                        q=q->next;
                                }
                                h2.next=this->next;
                                this->next=NULL;
                                this->add(h1,h2);
                                p=p->next;
                        }
                }

};


main()
{
        polynomial<int> h1,h2,h3;
        cout<<"Enter the First Polynomial :-";
        h1.create();
        cout<<"\nFirst Polynomial :-  ";
        h1.display();
        cout<<"\n\nEnter the Sceond Polynomial :-";
        h2.create();
        cout<<"\nSceond Polynomial :-  ";
        h2.display();
        h3.add(h1,h2);
        cout<<"\n\n\nAfter H1 + H2 :-   ";
        h3.display();
        h3.mul(h1,h2);
        cout<<"\n\nAfter H1 * H2 :-   ";
        h3.display();
}
```

Output:-

SET 1:-

```
Enter the First Polynomial :-
Enter the coefficient =4
Enter the exponent =6
Do you want to continue (y/n) =y
```

```
Enter the cofficent =2
Enter the exponent =4
Do you want to continue (y/n) =y

Enter the cofficent =5
Enter the exponent =3
Do you want to continue (y/n) =n

First Polynomial :-  4X^6 + 2X^4 + 5X^3

Enter the Sceond Polynomial :-
Enter the cofficent =7
Enter the exponent =5
Do you want to continue (y/n) =y

Enter the cofficent =8
Enter the exponent =3
Do you want to continue (y/n) =n

Sceond Polynomial :-  7X^5 + 8X^3


After H1 + H2 :-   4X^6 + 7X^5 + 2X^4 + 13X^3

After H1 * H2 :-   28X^30 + 14X^20 + 32X^18 + 35X^15 + 16X^12 + 40X^9
--------------------------------
Process exited after 87.11 seconds with return value 0
Press any key to continue . . .
```

SET 2:-

```
Enter the First Polynomial :-
Enter the coefficient =5
Enter the exponent =3
Do you want to continue (y/n) =y

Enter the coefficient =6
Enter the exponent =2
Do you want to continue (y/n) =n

First Polynomial :-  5X^3 + 6X^2

Enter the Sceond Polynomial :-
Enter the coefficient =2
Enter the exponent =6
Do you want to continue (y/n) =y

Enter the coefficient =3
Enter the exponent =4
Do you want to continue (y/n) =y

Enter the coefficient =8
```

Enter the exponent =3
Do you want to continue (y/n) =y

Enter the coefficient =5
Enter the exponent =2
Do you want to continue (y/n) =n

Sceond Polynomial :- 2X^6 + 3X^4 + 8X^3 + 5X^2

After H1 + H2 :- 2X^6 + 3X^4 + 13X^3 + 11X^2

After H1 * H2 :- 10X^18 + 27X^12 + 40X^9 + 18X^8 + 73X^6 + 30X^4

Process exited after 92.31 seconds with return value 0
Press any key to continue . . .

PRECAUTIONS AND DISCUSSION:

DISCUSSION:

1. This program uses template type class. Template can be used as macro which is an approach to generic programming. Since a template can be defined with a parameter thus it can be replaced by a specific data type at time of actual use of function. Class poly and add2poly are declared as template type.

2. Block 'polynomial' is declared comprising of two data parts c and e and node pointer to track the polynomial. e is the degree of the polynomial whereas c is the coefficient of the polynomial

3. Here, Polynomial ADDITION and MULTIPLICATION is performed respectively.

PRECAUTIONS:

1. Each line is ended with ";"(semicolon) to maintain the format rules.
2. The closing parameter of the class is followed by a ";"(Semicolon)
3. Namespace and proper header files are included to ensure proper execution of the library functions.
4. An infinite loop is used for proper execution according to the user.
5. Proper exit conditions are given to avoid incorrect outputs.
6. The array input for binary search must be in sorted order(for this program it has to be in ascending order).
7. If else condition is used for better execution of the program.

Assignment No :- 3

Program Statement:-

Write a program in C++ to generate postfix expression from the infix from, the infix expression constructor of class.

Program Algorithm:-

Description:

This is a program where we convert a infix expression to the postfix expression. Here two data part one is stack and other one is top that is initialize by -1. The data part are in the private part and others things into the public part.

Steps:

Step 1: top← -1 [It points to the top of Stack where DATA exist]

Step 2: create a function **display()**
[This function display the stack in the monitor]

Step 2.1: print stack
Step 2.2: STOP

Step 3: create a function **push(c)**
[This is a function for storing element "c" into the "stack"]
Step 3.1: top←top+1
Step 3.2: stack[top] ← c
Step 3.3: STOP

Step 4: create a function **pop()**
[This is a function for popping elements from the stack. Here we return the top element of the stack if stack is empty then it's returns -1]
Step 4.1: IF(top=-1) then
return(-1)
[END of IF]
Step 4.2: ELSE
return(stack[top--])
[END of ELSE]
Step 4.3: STOP

Step 5: create a function **priority(c)**

[This is a function to check the priority of the operation .It's returns a integer value depends on the priority of that operation.]

Step 5.1: IF(c='(') then
return()
[END of IF]
Step 5.2: IF(c='+' OR c='-') then
return(1)
[END of IF]
Step 5.3: IF(c='*' OR c='/') then
return(2)
[END of IF]
Step 5.4: IF(c='^' OR c='$') then
return(3)
[END of IF]
Step 5.5: STOP

Step 6: create a function **postfix(*c)**
[This is a function to change the infix expression to the equivalent postfix expression. c is a character type pointer which points a string's base address]

Step 6.1: i<-0 [i is an integer type variable]
Step 6.2: Print"The Equivalent Postfix Expression="
Step 6.3: repeat Step 6.4 to Step 6.15 IF c does not point to '\0'
Step 6.4: IF c points to any alphanumeric character then
P[i++]=*c [P is a character type array]
[END of IF]
Step 6.5: ELSE
Step 6.6: IF c points to '(' then
CALL "push(*c)"
[END of IF]
Step 6.7: ELSE
Step 6.8: IF c points to ')' then
Step 6.9: repeat Step 6.10 until (CALL "pop()" ='(') holds
Step 6.10: P[i++]<-stack[top+1]
[END of LOOP]
[END of Step 6.8 IF]
Step 6.11: ELSE
Step 6.12: repeat Step 6.13 IF((CALL "priority(stack[top])")>=(CALL "priority(*c)"))
Step 6.13: P[i++]<-pop()
[END of LOOP]
Step 6.14: CALL "push(*c)"
[END of Step 6.11 ELSE]
[END of Step 6.7 ELSE]
Step 6.15: c<-c+1
[END of Step6.3 LOOP]
Step 6.16: repeat Step 6.17 IF(top!=-1) holds
Step 6.17: P[i++]=pop()
[END of LOOP]
Step 6.18: P[i]='\0'
Step 6.19: Print P
Step 6.20: STOP

Step 7: Creating the function **main()**

Step 7.1: print “Enter a post fix expression =”
read a
Step 7.2: create A(a)
Step 7.3: A.display

Program Source Code:-

```
// HEADER FILES
#include<iostream>
#include<ctype.h>
#include<string.h>

using namespace std;

class postfix
{
	private:
		char stack[20];
		int top=-1;
	public:
		// A function to push a element in the stack
		void push(char c)
		{
			stack[++top]=c;
		}

		// A function to pop a element from the Stack
		char pop()
		{
			if(top==-1)
			 return(-1);
			else
			 return(stack[top--]);
		}

		// A function to check the priority
		int priority(char c)
		{
			if(c=='(')
			 return(0);
			if(c=='+'||c=='-')
			 return(1);
			if(c=='*'||c=='/')
			 return(2);
			if(c=='^'||c=='$')
			 return(3);
		}
```

```
        // A function to change the Infix expression to the equivalent Postfix expression
        postfix(char *c)
        {
                char P[20];
                int i=0;
                while(*c!='\0')
                {
                        if(isalnum(*c))
                         P[i++]=*c;
                        else
                         if(*c=='(')
                          push(*c);
                         else
                          if(*c==')')
                           while(pop()!='(')
                                P[i++]=stack[top+1];
                          else
                          {
                                while(priority(stack[top])>=priority(*c))
                                 P[i++]=pop();
                                push(*c);
                           }
                        c++;
                }
                while(top!=-1)
                 P[i++]=pop();
                P[i]='\0';
                strcpy(stack,P);
        }

        void display()
        {
                printf("\nThe Equivalent Postfix Expression=");
                cout<<stack;
        }

};


main()
{
        char a[20];
        cout<<"Enter a Infix Expression =";
        cin>>a;
        postfix A(a);
        A.display();
}
```

Output:-

SET 1:-
Enter a Infix Expression =2+3-3*9

The Equivalent Postfix Expression=23+39*-

Process exited after 24.44 seconds with return value 0
Press any key to continue . . .

SET 2:-
Enter a Infix Expression =2-4+4/5*20^4+23*12

The Equivalent Postfix Expression=24-45/204^*+2312*+

Process exited after 42.53 seconds with return value 0
Press any key to continue . . .

PRECAUTIONS AND DISCUSSION:

DISCUSSION:

1. Stack is a data structure that follows LIFO(last in first out) format, where each and every elements are either pushed in or popped out.
2. For Infix expression to Postfix expression conversion we use STACK. We take elements one by one from left from infix expression and push it in STACK based on priority. Similarly, based on priority we pop the elements from the STACK and form the postfix expression.

PRECAUTIONS:

1. Each line is ended with ";"(semicolon) to maintain the format rules.
2. The closing parameter of the class is followed by a ";"(Semicolon)
3. Namespace and proper header files are included to ensure proper execution of the library functions.
4. An infinite loop is used for proper execution according to the user.
5. Proper exit conditions are given to avoid incorrect outputs.
6. The array input for binary search must be in sorted order(for this program it has to be in ascending order).
7. If else condition is used for better execution of the program.

Assignment No :- 4

Program Statement:-

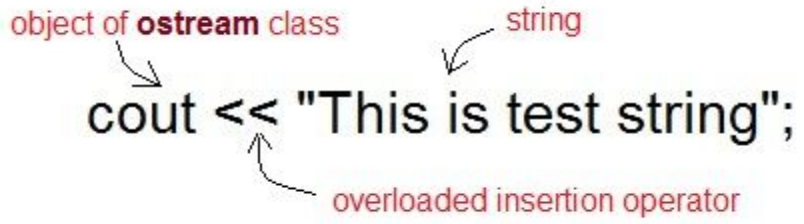
Write an object oriented program using C++ create vector type object .These vector object are created using singly link list type object which are again created using node multi-field object .Link list is used to incorporate the dynamic nature ,Every object may different type of numeric data values (use template).Each object is created with their operation ,in addition do the followings for the vector object :

a) Addition of two vector using function and operator overloading.
b) Subtraction of two vectors using function and operator overloading.
c) Scalar Multiplication of two vector using function and operator overloading.
d) Vector multiplication of two vector using function and operator overloading
e) Measuring length of the vector

Program Algorithm:-

Description:

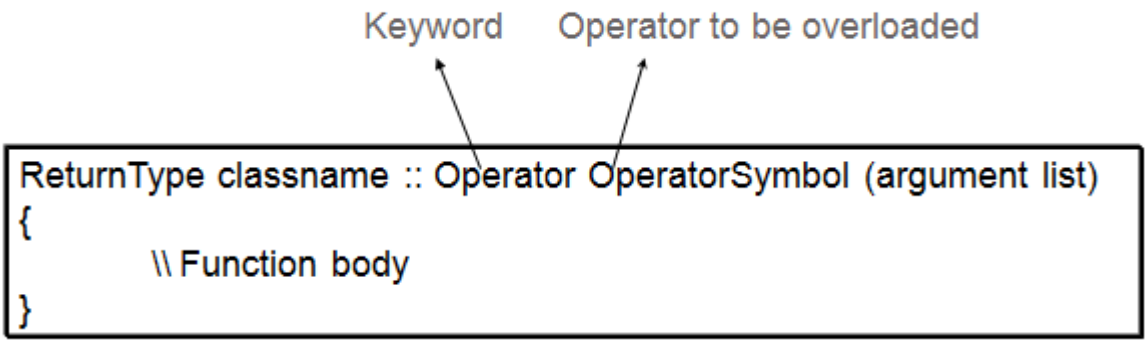
Operator overloading is an important concept in C++. It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operation on user-defined data type. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc.



Almost any operator can be overloaded in C++. However there are few operator which can not be overloaded. **Operator that are not overloaded** are follows

- scope operator - ::
- sizeof
- member selector - .
- member pointer selector - *
- ternary operator - ?:

Operator Overloading Syntax



Implementing Operator Overloading

Operator overloading can be done by implementing a function which can be :

1. Member Function
2. Non-Member Function
3. Friend Function

Operator overloading function can be a member function if the Left operand is an Object of that class, but if the Left operand is different, then Operator overloading function must be a non-member function.

Operator overloading function can be made friend function if it needs access to the private and protected members of class.

Restrictions on Operator Overloading

Following are some restrictions to be kept in mind while implementing operator overloading.

1. Precedence and Associativity of an operator cannot be changed.
2. Arity (numbers of Operands) cannot be changed. Unary operator remains unary, binary remains binary etc.
3. No new operators can be created, only existing operators can be overloaded.
4. Cannot redefine the meaning of a procedure. You cannot change how integers are added.

Steps:

Step 1: Create two class, one for vector addition, subtraction, multiplication and another for polynomial insertion and print using operator overloading.
Step 2: Overload '+', '-', and '*' operator for vector
Step 3: Overload '<<' and '>>' operator to take a input of a polynomial and print output of a polynomial.
Step 4: Ask the user to choose between Subtraction of the Vector or Sum of Vectors or Product of Vectors or Product of scalar and vector or Insert a polynomial or Display the polynomial or Length of vector or exit.
Step 5: If user enters exit then go to step 8 else, Call the suitable functions by objects.
Step 6: Display the result.
Step 7: Go to step 4.
Step 8: STOP

Program Source Code:-

```
// Hrader File
#include<iostream>

using namespace std;

template<class T>class Vector
{
	private:
		T x;
		Vector *next;
	public:
		Vector(T a=0)
		{
			x=a;
			next=NULL;
		}

		void create()
		{
			Vector *p,*q;
			char ch='Y';
			T x;
			this->next=NULL;
			while(ch=='y'||ch=='Y')
			{
				cout<<"\nEnter a value=";
				cin>>x;
				p=new Vector(x);
				if(this->next==NULL)
				 this->next=p;
				else
				 q->next=p;
				q=p;
				cout<<"Do you want to continue(Y/N)=";
				cin.sync();
				cin>>ch;
			}
		}

		void display()
		{
			Vector *p;
			p=this->next;
			cout<<"{ ";
			while(p!=NULL)
			{
				cout<<" "<<p->x<<" ";
				p=p->next;
			}
			cout<<" }";
		}
```

```
int length()
{
        if(this->next==NULL)
         return 0;
        Vector *p;
        p=this->next;
        int i=0;
        while(p!=NULL)
        {
                p=p->next;
                i++;
        }
        return i;
}

Vector operator +(Vector b)
{
        Vector h,*p,*q,*r,*t;
        p=this->next;
        q=b.next;
        while(p!=NULL and q!=NULL)
        {
                r=new Vector(p->x+q->x);
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
                p=p->next;
                q=q->next;
        }
        while(p!=NULL)
        {
                r=new Vector(p->x);
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
                p=p->next;
        }
        while(q!=NULL)
        {
                r=new Vector(q->x);
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
                q=q->next;
```

```
        }
        return h;
}

Vector operator -(Vector b)
{
        Vector h,*p,*q,*r,*t;
        p=this->next;
        q=b.next;
        while(p!=NULL and q!=NULL)
        {
                r=new Vector(p->x-q->x);
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
                p=p->next;
                q=q->next;
        }
        while(p!=NULL)
        {
                r=new Vector(p->x);
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
                p=p->next;
        }
        while(q!=NULL)
        {
                r=new Vector(-q->x);
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
                q=q->next;
        }
        return h;
}

T operator *(Vector b)
{
        T sum=0;
        Vector *p,*q;
        p=this->next;
        q=b.next;
        while(p!=NULL and q!=NULL)
        {
```

```
                        sum=sum+p->x*q->x;
                        p=p->next;
                        q=q->next;
                }
                return sum;
        }
};


main()
{
        Vector<int> a,b,c;
        cout<<"Enter Vector A :-\n";
        a.create();
        cout<<"\nVector A :-  ";
        a.display();
        cout<<"\n\nLength of Vector A ="<<a.length();
        cout<<"\n\nEnter Vector B :-\n";
        b.create();
        cout<<"\nVector B :-  ";
        b.display();
        cout<<"\n\nLength of Vector B ="<<b.length();
        cout<<"\n\nAfter C=A+B :-\n";
        c=a+b;
        cout<<"\nVector C :-  ";
        c.display();
        cout<<"\n\nLength of Vector C ="<<c.length();
        cout<<"\n\nAfter C=A-B :-\n";
        c=a-b;
        cout<<"\nVector C :-  ";
        c.display();
        cout<<"\n\nLength of Vector C ="<<c.length();
        cout<<"\n\nScalar Value of A.B ="<<a*b;;
}
```

Output:-

SET 1:-

```
Enter Vector A :-

Enter a value=2
Do you want to continue(Y/N)=y

Enter a value=8
Do you want to continue(Y/N)=y

Enter a value=4
Do you want to continue(Y/N)=n

Vector A :-  {  2  8  4  }
```

```
Length of Vector A =3

Enter Vector B :-

Enter a value=4
Do you want to continue(Y/N)=n

Vector B :- { 4 }

Length of Vector B =1

After C=A+B :-

Vector C :- { 6 8 4 }

Length of Vector C =3

After C=A-B :-

Vector C :- { -2 8 4 }

Length of Vector C =3

Scalar Value of A.B =8
--------------------------------
Process exited after 9.59 seconds with return value 0
Press any key to continue . . .
```

SET 2:-

```
Enter Vector A :-

Enter a value=6
Do you want to continue(Y/N)=y

Enter a value=3
Do you want to continue(Y/N)=y

Enter a value=2
Do you want to continue(Y/N)=y

Enter a value=8
Do you want to continue(Y/N)=n

Vector A :- { 6 3 2 8 }

Length of Vector A =4

Enter Vector B :-

Enter a value=3
```

```
Do you want to continue(Y/N)=y

Enter a value=1
Do you want to continue(Y/N)=n

Vector B :- { 3 1 }

Length of Vector B =2

After C=A+B :-

Vector C :- { 9 4 2 8 }

Length of Vector C =4

After C=A-B :-

Vector C :- { 3 2 2 8 }

Length of Vector C =4

Scalar Value of A.B =21
--------------------------------
Process exited after 17.56 seconds with return value 0
Press any key to continue . . .
```

PRECAUTIONS AND DISCUSSION:

DISCUSSION:

1. Vector is a geometric object that has magnitude and direction. Vectors can be added to another vector according to vector algebra. In this program , we overload the operator to perform vector addition, subtraction, vector multiplication as well as scalar multiplication.
2. In case of operator overloading, the object on the right hand side of the operator is always assumed as argument by compiler.

PRECAUTIONS:

1. Each line is ended with ";"(semicolon) to maintain the format rules.
2. The closing parameter of the class is followed by a ";"(Semicolon)
3. Namespace and proper header files are included to ensure proper execution of the library functions.
4. An infinite loop is used for proper execution according to the user.
5. Proper exit conditions are given to avoid incorrect outputs.
6. The array input for binary search must be in sorted order(for this program it has to be in ascending order).
7. If else condition is used for better execution of the program.

Assignment No :- 5

Program Statement:-

Design a class complex having two float type data members real and imaginary representing the real and imaginary parts of a complex number respectively. Write suitable constructors to set the value of the objects. Overload the +,-,*, / operators to perform addition, subtraction, multiplication and division operation respectively on the complex objects

Program Algorithm:-

Description:

The operator overloading feature in C++ allows us to operate upon objects of a class, comprising of one or more data type. Hence, it provides simplicity to program. We can overload unary and binary operators. Ternary operator overloading is not possible. e.g.: +, -, *, /, =, +=, -=, ++, --, !=... etc.; these operators can be overloaded.
In the complex class, we overload the four basic arithmetic operators to work on complex numbers given as input. We use the following formulae to implement these operators:

$$(a + bi) + (c + di) = (a + c) + (b + d)\,i$$
$$(a + bi) - (c + di) = (a - c) + (b - d)\,i$$
$$(a + bi) * (c + di) = (ac - bd) + (bc + ad)$$
$$(a + bi)/(c + di) = \frac{ac + bd}{c^2 + d^2} + \left(\frac{bc - ad}{c^2 + d^2}\right) i$$

Steps:

Algorithm complex:: float real, float imaginary

Step 1: Declare various operations for complex class to perform.

Step 2 a: [Initialize float numbers]
Set real and imaginary to data given by user.

Step 2 b: [Addition]
/*First number is stored in a and the second number is stored in b.*/
/* we use the formulae $(a + bi) + (c + di) = (a + c) + (b + d)\,i$ to calculate the sum of the complex numbers*/
Set temp equal to the sum of the numbers.
Return temp to the calling object.

Step 2 c: [Subtraction]
/*First number is stored in a and the second number is stored in b.*/
/* we use the formulae $(a + bi) - (c + di) = (a - c) + (b - d)\,i$ to calculate the difference of the complex numbers*/
Set temp equal to the difference of the numbers.
Return temp to the calling object.

Step 2 d: [Multiplication]
/*First number is stored in a and the second number is stored in b.*/

/* we use the formulae $(a + bi) * (c + di) = (ac - bd) + (bc + ad)\, i$ to calculate the product of the complex numbers*/
Set temp equal to the product of the numbers.
Return temp to the calling object.

Step 2 e: [Division]
/*First number is stored in a and the second number is stored in b.*/
/* we use the formulae to calculate the product of $(a + bi)/(c + di) = \frac{ac+bd}{c^2+d^2} + \left(\frac{bc-ad}{c^2+d^2}\right) i$the complex numbers*/
Set temp equal to the dividend of the numbers.
Return temp to the calling object.

Step 2 f: [Displaying the numbers]
Print both numbers.

Step 3: [END]

Program Source Code:-

```
#include<iostream.h>
#include<conio.h>
class complex
{
  float real,img;
  public:
   complex()
   {
     real=0;
     img=0;
    }
    complex(float a,float b)
    {
      real=a;
      img=b;
    }
    void getdata()
    {
      cout<<"\nEnter the value of the real and imaginary  respectively :\n";
      float a;
      float b;
      cin>>a>>b;
 real=a;
 img=b;
}
 void display()
{
 cout<<"\nThe complex number is :\n"<<real<<" + "<<img<<"i";
}
 friend complex operator +(complex obj1,complex obj2)
{
```

```
 complex temp;
  temp.real=obj1.real + obj2.real;
  temp.img=obj1.img + obj2.img;
   return(temp);
}
friend complex operator -(complex obj1,complex obj2)
{
  complex temp;
  temp.real=obj1.real - obj2.real;
  temp.img=obj1.img - obj2.img;
  return(temp);
}

friend complex operator *(complex obj1,complex obj2)
{
 complex temp;
 temp.real=obj1.real * obj2.real - obj1.img * obj2.img ;
 temp.img=obj1.img * obj2.real + obj1.real * obj2.img;
 return(temp);
}

friend complex operator /(complex obj1,complex obj2)
{
 complex temp;
 temp.real=(obj1.real * obj2.real + obj1.img * obj2.img) /  (obj2.real * obj2.real + obj2.img *
obj2.img);
 temp.img=(obj1.img * obj2.real - obj1.real * obj2.img) / (obj2.real * obj2.real + obj2.img *
obj2.img);
 return(temp);
}
        };

int main()
{
  complex a,b,c,d,e,f;
  cout<<"\nEnter two complex numbers\n";
  a.getdata();
  a.display();
  b.getdata();
  b.display();
  cout<<"\nBasic operations are as follows:\n";
  cout<<"\nValue after addition is:\t";
   c=a+b;
  c.display();
  cout<<"\nValue after subtraction is:\t";
   d=a-b;
  d.display();
  cout<<"\nValue after multipication is:\t";
   e=a*b;
  e.display();
  cout<<"\nValue after division is:\t";
```

```
   f=a/b;
  f.display();
  getch();
}
```

Output:-

```
Enter two complex numbers
Enter the value of the real and imaginary respectively :
2
3

The complex number is :
2 + 3 i
Enter the value of the real and imaginary respectively :
3
6

The complex number is :
3 + 6 i
Basic operations are as follows:

Value after addition is:
The complex number is :
5 + 9 i
Value after subtraction is:
The complex number is :
-1 + -3 i
Value after multipication is:
The complex number is :
-12 + 21 i
Value after division is:
The complex number is :
0 + 0 i

Enter two complex numbers

Enter the value of the real and imaginary respectively :
2.3
4.5

The complex number is :
2.3 + 4.5 i
Enter the value of the real and imaginary respectively :
3.4
6.5

The complex number is :
3.4 + 6.5 i
```

```
Basic operations are as follows:
Value after addition is:
The complex number is :
5.7 + 11 i
Value after subtraction is:
The complex number is :
-1.1 + -2 i
Value after multipication is:
The complex number is :
-21.43 + 30.25 i
Value after division is:
The complex number is :
        0.688905 + 0.006504 i
--------------------------------
Process exited after 2.802 seconds with return value 0
Press any key to continue . . .
```

DISCUSSION:-

In the program, we have designed a procedure to overload +, -, * and / operators.
The program can execute with complex number system, i.e. it can add, subtract, multiply and divided two complex number at once.

Assignment No :- 6

Program Statement:-

Write a program to create a class DLLIST to implement doubly linked list of template type. The program should have support for ordered insertion and deletion of a node.

Program Algorithm:-

Description:

A template can be used to create a family of classes or Function. A class template for an array class would enable us to create array of type such as integer array and float array. When object of a specific type is defined for actual use the template definition of for that class is substituted with the required data type. Since a template is defined with the parameter that would be replaced by the specific data type at the time of the actual use of the class function the template are sometimes called parameterized classes.

Steps:

Step 1: T is the template class which has three member function getdata(),creatlist() and displaylist() declared inside the public mode of the class and defined outside the class by using scope resolution operator.

Step 2: Inside the getdata() function the value is taken by the user and call by value by another member function and head is passed as reference and head is initialized as NULL.

Step 3: In creatlist() member function new1 node is created and initialized value passed from getdata() function,
new1->left=NULL and new1->right=NULL.

Step 4: If head->left=NULL then head->left=new1 and
head->right=new1.
Else head->right->right=new1;
new1->left=head->right;
head->right=new1;

Step 5: In displaylist member function ptr is pointed on head and repeat while ptr->left!=NULL and initialized ptr=head->right
a) print the data by ptr->data
b) ptr=ptr->left
[End of while loop]

Step 6: All member function is accessible by the object of T class .

Step 7: End.

Program Source Code:-

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
```

```
template<class T>
class node
{
  T data;
  node<T> *prev;
  node<T> *next;
  public:
  void create(node<T> *&head);
  void display(node<T> *&head);
  void del(node<T> *&head);
  node<T>* getnode(T);
  int count(node<T> *&head);
};
template<class T>
node<T>* node<T>::getnode(T x)
{
  node<T> *ptr;
  ptr=new node;
  ptr->next=NULL;
  ptr->data=x;
  ptr->prev=NULL;
  return (ptr);
}
template<class T>
int node<T>::count(node<T> *&head)
{
   int c=0;
   node *ptr;
   ptr=head;
   while(ptr!=NULL)
   {
      ptr=ptr->next;
      c++;
   }
   c=c/2;
   return c;
}
template<class T>
void node<T>::display(node<T> *&head)
{
  node *ptr;
  ptr=head;
  while(ptr!=NULL)
  {
    cout<<"   "<<ptr->data;
    ptr=ptr->next;
  }
}
template<class T>
void node<T>::create(node<T> *&head)
{
```

```
    T x;
    node<T> *temp,*tr,*tr1;
    cout<<"\n\tEnter value (-999 to stop): ";
    cin>>x;
    if(x!=-999)
    {
      while(x!=-999)
      {
           temp=getnode(x);
           if(head==NULL)
             head=temp;
           else
           {
             tr=head;
             tr1=head;
             while(tr!=NULL)
             {
                if(tr->data<temp->data)
                {
                     tr1=tr;
                     tr=tr->next;
                }
                else
                  break;
             }
            if(tr1==tr)
             {
               temp->next=head;
               head=temp;
             }
             else if(tr->data>=temp->data)
             {
               temp->prev=tr1;
               tr1->next=temp;
               temp->next=tr;
               tr->prev=temp;
             }
             else
             {
                tr1->next=temp;
                temp->prev=tr1;
             }

           }
           cout<<"\n\tEnter value (-999 to stop): ";
           cin>>x;
      }
    }
}
template<class T>
void node<T>::del(node<T> *&head)
```

```
{
 T x;
 node<T> *tr,*tr1;
 int y=1;

 while(y!=0)
 {
   if(head==NULL)
 {
  cout<<"\n\tList is empty.....";
  getch();
  exit(0);
 }
 cout<<"\n\tEnter value for deletion: ";
 cin>>x;
 tr1=tr=head;
 while(tr!=NULL)
 {
  if(tr->data!=x)
  {
    tr1=tr;
    tr=tr->next;
  }
  else
    break;
 }
 if(tr==NULL)
 {
  cout<<"\n\tValue not found.....";
 }
 else
 {
  if(tr1==tr)
  {
    head=head->next;
    head->prev=NULL;
  }
  else
  {
    tr1->next=tr->next;
    tr1->next->prev=tr1;
  }
  cout<<"\n\t"<<tr->data<<" is deleted.....";
  delete tr;
  cout<<"\n\tNow the list is.....";
  display(head);
 }
  cout<<"\n\tDo you want to delete another record (1 for yes & 0 for no): ";
  cin>>y;
 }
}
```

```
void main()
{
  node<int> *head,obj;
  head=NULL;
  clrscr();
  obj.create(head);
  cout<<"\n\tOriginal list is.....";
  obj.display(head);
  obj.del(head);
  getch();
}
```

Output:-

```
Enter value (-999 to stop): 2

 Enter value (-999 to stop): 5

 Enter value (-999 to stop): 8

 Enter value (-999 to stop): 1

 Enter value (-999 to stop): -999

 Original list is.....   1   2   5   8
 Enter value for deletion: 5

 5 is deleted.....
 Now the list is.....   1   2   8
 Do you want to delete another record (1 for yes & 0 for no): 0
```

DISCUSSION:-

A class created from class template is called template class. The syntax for definition of an object of a template class is
Classname<type>objectname (arglist).
The array is initialized by the constructor. Here the member functions are defined inside the public visibility labels such that it is accessible by the object of the associated class declared inside the main function. The member function can wrap the private data from the outside function and we can call another member function from another member function. Every member function that is defined outside the template class follows that syntax….
Returntypeclassname<T> :: function name(arglist)
and it is preceded by this template<class T>.

Assignment No :- 7

Program Statement:-

Write a program manipulating linked list supporting node operations as follows:
Node=node+2;
Node=node-3;
Node<int>*n=node1+node2;

The first statement creates a new node with node information 2 and the second statement deletes a node with node information 3, the node class must be of type template.

Program Algorithm:-

Description:

A linked list is a linear data structure where each element is a separate object. Linked list elements are not stored at contiguous location; the elements are linked using pointers. Each node of a list is made up of two items - the data and a reference to the next node. The last node has a reference to null.

Steps:

```
STEP 1:         Define a generic class named node as follows:
    1.1     Define private data members x and a pointer next
    1.2     Define all member procedures as pubic
    1.3     PROCEDURE node(a = 0)
    1.3.1       Set x = a
    1.3.2       Set next = NULL
            End of Procedure
    1.4     PROCEDURE operator >>(input stream object in, node object h)
    1.4.1       Define 2 pointers of type node: p,q
    1.4.2       Set ch='Y'
    1.4.3       Set h.next = NULL
    1.4.4       Repeat while ch='Y' or ch='y'
    1.4.4.1         Print "Enter a value: "
    1.4.4.2         Input x
    1.4.4.3         Set p = new object node(x)
    1.4.4.4         If (h.next = NULL) then
                        h.next = p
                    Else
                        q->next = p
                    End of If
    1.4.4.5         Set q = p
    1.4.4.6         Print "Do you want to continue? (Y/N) "
    1.4.4.7         Input ch
```

```
                End of Loop
        End of Procedure
1.5     PROCEDURE operator << (output stream object out, node object h)
1.5.1           If (h.next = NULL)
                        Print “List doesn’t exist”
                        Return from Procedure
                End of If
1.5.2           Set p = h.next
1.5.3           Print “The list: “
1.5.4           Repeat while p not equal to NULL
                        Print p->x
                        Set p = p->next
                End of Loop
        End Procedure
1.6     PROCEDURE operator + (x)
1.6.1           Set c = 1
1.6.2           If (object pointer -> next = NULL)
                        Print “List doesn’t exist”
                        Return value h
                End of If
1.6.3           Set p = object pointer -> next
1.6.4           Repeat while p is not equal to NULL
1.6.4.1                 Set q = new object node(p->x)
1.6.4.2                 If (h.next = NULL)
                                Set h.next = q
                        Else
                                Set t->next = q
                        End of If
1.6.4.3                 Set t = q
1.6.4.4                 Set p = p->next
                End of Loop
1.6.5           Print “Enter position: “
1.6.6           Input pos
1.6.7           Set p = new object node(x)
1.6.8           If pos = 1 then
                        Set p->next = h.next
                        Set h.next = p
                        Return value h
                End of If
1.6.9           Set q = h.next
1.6.10          Repeat while (q->next not equal to NULL and c < pos -1)
                        Set q = q->next
                        Set c = c+1
                End of Loop
1.6.11          Set p->next = q->next
1.6.12          Set q->next = p
1.6.13          Return value h
        End of Procedure
1.7     PROCEDURE operator –(x)
1.7.1           If (object pointer ->next = NULL) then
                        Print “List doesn’t exist”
```

```
                        Return value h
                    End If
        1.7.2       Set p = object pointer -> next
        1.7.3       Repeat while p is not equal to NULL
        1.7.3.1             q = create new object node(p->x)
        1.7.3.2             If (h.next = NULL)
                                Set h.next = q
                            Else
                                Set t->next = q
                            End of If
        1.7.3.3             Set t=q
        1.7.3.4             Set p = p->next
                    End of Loop
        1.7.4       Set p = h.next
        1.7.5       If (p->x = x)
                            h.next = p->next
                            Print “Deletion complete”
                            Delete memory pointed to by p
                            Return value h
                    End of If
        1.7.6       Set q = h.next
        1.7.7       While ( q is not equal to NULL and q->x is not equal to x)
                            Set p = q
                            Set q = q->next
                    End of Loop
        1.7.8       If (q = NULL) then
                            Print “Element not found”
                    Else
                            Set p->next = q->next
                            Print “Deletion complete”
                            Delete memory pointed to by p
                    End of If
                End of Procedure
                End of class definition
STEP 2:             Declare an object of class node named h of type integer
STEP 3:             Repeat STEP 4 to STEP 6 forever
STEP 4:                 Print “1. Create list”
                    Print “2. Insert element at any position”
                    Print “3. Delete element from any position”
                    Print “4. Display the list”
                    Print “5. Exit”
                    Print “Enter your choice: “
STEP 5:                 Input n
STEP 6:     6.1         If n = 1 then
        6.1.1               Input h
        6.2         Else, if n = 2 then
        6.2.1               Print “Enter an element to insert: “
        6.2.2               Input x
        6.2.3               H = h + x
        6.3         Else, if n = 3 then
        6.3.1               Print “Enter an element to delete: “
```

```
        6.3.2                   Input x
        6.3.3                   h = h - x
        6.4             Else, if n = 4 then
        6.4.1                   Print h
        6.5             Else
        6.5.1                   Exit from loop
                        End of If
                End of Loop
STEP 7:                 STOP
```

Program Source Code:-

```
// Header File
#include<iostream>

using namespace std;

template<class T>
class node
{
        private:
                T x;
                node *next;
        public:
                node(T a=0)
                {
                        x=a;
                        next=NULL;
                }

                friend void operator >>(istream &in,node &h)
                {
                        node *p,*q;
                        char ch='Y';
                        T x;
                        h.next=NULL;
                        while(ch=='y'||ch=='Y')
                        {
                                cout<<"\nEnter a value=";
                                cin>>x;
                                p=new node(x);
                                if(h.next==NULL)
                                 h.next=p;
                                else
                                 q->next=p;
                                q= p;
                                cout<<"Do you want to continue(Y/N)=";
                                cin.sync();
                                cin>>ch;
                        }
```

```
}

friend void operator <<(ostream &out,node &h)
{
        node *p;
        if(h.next==NULL)
        {
                cout<<"\nList Not Exist.";
                return;
        }
        p=h.next;
        cout<<"\nThe List :-\n";
        while(p!=NULL)
        {
                cout<<p->x<<"\t";
                p=p->next;
        }
}

node operator +(T x)
{
        node *p,*q,*t,h;
        int pos,c=1;
        if(this->next==NULL)
        {
                cout<<"\nList Not Exist.";
                return h;
        }
        p=this->next;
        while(p!=NULL)
        {
                q=new node(p->x);
                if(h.next==NULL)
                 h.next=q;
                else
                 t->next=q;
                t=q;
                p=p->next;
        }
        cout<<"Enter the Position=";
        cin>>pos;
        p=new node(x);
        if(pos==1)
        {
                p->next=h.next;
                h.next=p;
                return h;
        }
        q=h.next;
        while(q->next!=NULL&&c<pos-1)
        {
```

```
                q=q->next;
                c=c+1;
            }
            p->next=q->next;
            q->next=p;
            return h;
        }

        node operator -(T x)
        {
            node *p,*q,*t,h;
            if(this->next==NULL)
            {
                cout<<"\nList Not Exist.";
                return h;
            }
            p=this->next;
            while(p!=NULL)
            {
                q=new node(p->x);
                if(h.next==NULL)
                 h.next=q;
                else
                 t->next=q;
                t=q;
                p=p->next;
            }
            p=h.next;
            if(p->x==x)
            {
                h.next=p->next;
                cout<<"Deletion Complete.";
                delete p;
                return h;
            }
            q=h.next;
            while(q!=NULL&&q->x!=x)
            {
                p=q;
                q=q->next;
            }
            if(q==NULL)
             cout<<"\nElement not found.";
            else
            {
            p->next=q->next;
                cout<<"Deletion Complete.";
                delete q;
            }
        }
};
```

```
main()
{
	node<int> h;
	int n,x;
	while(1)
	{
		cout<<"\n\n1 for Create List.";
		cout<<"\n2 for Insert Element at any Position.";
		cout<<"\n3 for Delete Element from any Element.";
		cout<<"\n4 for Display the list.";
		cout<<"\n0 for Exit.";
		cout<<"\nEnter Your Choice=";
		cin>>n;
		switch(n)
		{
			case 1:cin>>h;
				break;
			case 2:cout<<"\nEnter A Element to insert =";
				cin>>x;
				h=h+x;
				break;
			case 3:cout<<"\nEnter A Element to Delete =";
				cin>>x;
				h=h-x;
				break;
			case 4:cout<<h;
				break;
			case 0:exit(0);
		}
	}
}
```

Output:-

```
1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=1

Enter a value=6
Do you want to continue(Y/N)=y

Enter a value=4
```

```
Do you want to continue(Y/N)=y

Enter a value=2
Do you want to continue(Y/N)=y

Enter a value=7
Do you want to continue(Y/N)=y

Enter a value=12
Do you want to continue(Y/N)=y

Enter a value=9
Do you want to continue(Y/N)=y

Enter a value=3
Do you want to continue(Y/N)=n


1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=4

The List :-
6       4       2       7       12      9       3

1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=3

Enter A Element to Delete =7
Deletion Complete.

1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=3

Enter A Element to Delete =7

Element not found.

1 for Create List.
2 for Insert Element at any Position.
```

```
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=4

The List :-
6       4       2       12      9       3

1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=3

Enter A Element to Delete =12
Deletion Complete.

1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=4

The List :-
6       4       2       9       3

1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=2

Enter A Element to insert =12
Enter the Position=1


1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=4

The List :-
12      6       4       2       9       3

1 for Create List.
2 for Insert Element at any Position.
```

```
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=2

Enter A Element to insert =20
Enter the Position=20


1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=4

The List :-
12      6       4       2       9       3       20

1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=2

Enter A Element to insert =34
Enter the Position=5


1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=4

The List :-
12      6       4       2       34      9       3       20

1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=3

Enter A Element to Delete =2
Deletion Complete.

1 for Create List.
```

```
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=4

The List :-
12      6       4       34      9       3       20

1 for Create List.
2 for Insert Element at any Position.
3 for Delete Element from any Element.
4 for Display the list.
0 for Exit.
Enter Your Choice=0

--------------------------------
Process exited after 115.7 seconds with return value 0
Press any key to continue . . .
```

PRECAUTIONS AND DISCUSSION:

DISCUSSION:

1. The principal benefit of a linked list over a conventional array is that the list elements can easily be inserted or removed without reallocation or reorganization of the entire structure because the data items need not be stored contiguously in memory or on disk, while an array has to be declared in the source code, before compiling and running the program. Linked lists allow insertion and removal of nodes at any point in the list, and can do so with a constant number of operations if the link previous to the link being added or removed is maintained during list traversal.
2. Linked lists are a dynamic data structure, allocating the needed memory while the program is running.
3. Insertion and deletion node operations are easily implemented in a linked list.
4. Linear data structures such as stacks and queues are easily executed with a linked list.
5. They can reduce access time and may expand in real time without memory overhead.

PRECAUTIONS:

1. Each line is ended with ";"(semicolon) to maintain the format rules.
2. The closing parameter of the class is followed by a ";"(Semicolon)
3. Namespace and proper header files are included to ensure proper execution of the library functions.
4. An infinite loop is used for proper execution according to the user.
5. Proper exit conditions are given to avoid incorrect outputs.
6. The array input for binary search must be in sorted order(for this program it has to be in ascending order).

7. If else condition is used for better execution of the program.

Assignment No :- 8

Program Statement:-

Write functions to perform matrix operations: (i) Addition (ii) Subtraction (iii) Multiplication (iv) Transpose

Program Algorithm:-

Description:

In mathematics, a matrix is a rectangular arrayof numbers, symbols, or expressions, arranged in rows and columns.
Provided that they have the same size (each matrix has the same number of rows and the same number of columns as the other), two matrices can be added or subtracted element by element. The rule for matrix multiplication, however, is that two matrices can be multiplied only when the number of columns in the first equals the number of rows in the second (i.e., the inner dimensions are the same, n for an (m×n)-matrix times an (n×p)-matrix, resulting in an (m×p)-matrix.

Steps:

```
STEP 1:          PROCEDURE mat_add(mat1[ ][ ], mat2[ ][ ], row1, col1, row2, col2, res[ ]
[ ])
    1.1          If (row1 not equal to row2 OR col1 not equal to col2) then
                     Print “Addition not possible due to different dimensions”
                     Return from procedure
                 End of If
    1.2          Set i = 0, j  = 0
    1.3          Repeat until i>= row1
                     Repeat until j >= col1
                         res[i][j] = mat1[i][j] + mat2[i][j]
                         Set j = j + 1
                     End of Loop
                     Set i = i + 1
                 End of Loop
    1.4          Print “Result: “
    1.5          Set i = 0, j  = 0
    1.6          Repeat until i>= row1
                     Repeat until j >= col1
                         Print res[i][j]
                         Set j = j + 1
                     End of Loop
                     Set i = i + 1
                 End of Loop
             End of Procedure
```

```
STEP 2:         PROCEDURE mat_sub(mat1[ ][ ], mat2[ ][ ], row1, col1, row2, col2, res[ ]
                [ ])
    2.1         If (row1 not equal to row2 OR col1 not equal to col2) then
                    Print “Subtraction not possible due to different dimensions”
                    Return from procedure
                End of If
    2.2         Set i = 0, j  = 0
    2.3         Repeat until i>= row1
                    Repeat until j >= col1
                        res[i][j] = mat1[i][j] - mat2[i][j]
                        Set j = j + 1
                    End of Loop
                    Set i = i + 1
                End of Loop
    2.4         Print “Result: “
2.5         Set i = 0, j  = 0
    2.6         Repeat until i>= row1
                    Repeat until j >= col1
                        Print res[i][j]
                        Set j = j + 1
                    End of Loop
                    Set i = i + 1
                End of Loop
            End of Procedure
STEP 3:         PROCEDURE mat_add(mat1[ ][ ], mat2[ ][ ], row1, col1, row2, col2, res[ ]
[ ])
    3.1         If (col1 not equal to row2) then
                    Print “Multiplication not possible”
                    Return from procedure
                End of If
    3.2         Set i = 0, j  = 0, k=0
    3.3         Repeat until i>= row1
                    Repeat until j >= col2
                        Repeat until k>= col1
                            res[i][j] = res[i][j]+(mat1[i][k] * mat2[k][j])
                            Set k = k + 1
                        End of Loop
                        Set j = j + 1
                    End of Loop
                    Set i = i + 1
                End of Loop
    3.4         Print “Result: “
3.5         Set i = 0, j  = 0
    3.6         Repeat until i>= row1
                    Repeat until j >= col2
                        Print Res[i][j]
                        Set j = j + 1
                    End of Loop
                    Set i = i + 1
                End of Loop
            End of Procedure
```

```
STEP 4:          PROCEDURE mat_trans(mat[ ][ ], row, col, res[ ] [ ])
    4.1          Set i = 0, j  = 0
    4.2          Repeat until i>= row1
                     Repeat until j >= col1
                         Res[j][i] = mat[i][j]
                         Set j = j + 1
                     End of Loop
                     Set i = i + 1
                 End of Loop
    4.3          Print “Result: “
4.4          Set i = 0, j  = 0
    4.5          Repeat until i>= row
                     Repeat until j >= col
                         Print Res[i][j]
                         Set j = j + 1
                     End of Loop
                     Set i = i + 1
                 End of Loop
             End of Procedure
STEP 5:          Set ch = ‘n’
STEP 6:          Print “Enter number of rows and columns of first matrix: ”
STEP 7:      Input row1,col1
STEP 7:      Print “Enter number of rows and colums of second matrix: ”
STEP 8:          Input row2,col2
STEP 9:          Print “Enter elements of first matrix: ”
STEP 10:     Set i = 0, j  = 0
STEP 11:     Repeat until i>= row
                 Repeat until j >= col
                     Input A[i][j]
                     Set j = j + 1
                 End of Loop
                 Set i = i + 1
             End of Loop
STEP 12:     Print “Enter elements of second matrix: ”
STEP 13:     Set i = 0, j  = 0
STEP 14:     Repeat until i>= row
                 Repeat until j >= col
                     Input B[i][j]
                     Set j = j + 1
                 End of Loop
                 Set i = i + 1
             End of Loop
STEP 15:     Print “1. Matrix Addition”
             Print “2. Matrix Subtraction”
             Print “3. Matrix Multiplication”
             Print “4. Matrix Transpose”
             Print “5. Exit”
STEP 16:     Input choice
STEP 17:     If choice = 1 then
                 Call procedure mat_add(A, B, row1, col1, row2, col2, result)
             Else, if choice = 2 then
```

```
                Call procedure mat_sub(A, B, row1, col1, row2, col2, result)
            Else, if choice = 3 then
                Call procedure mat_mul (A, B, row1, col1, row2, col2, result)
            Else, if choice = 4 then
                Call procedure mat_trans(A, row1, col1, result)
                Call procedure mat_trans(B, row2, col2, result)
            Else
                Exit from program
            End of If
STEP 18:    Print “Do you want to continue? (Y/N)”
STEP 19:    Input ch
STEP 20:    Repeat STEP 15 to STEP 19 while ch = ‘y’ OR ch=’Y’
STEP 21:    STOP
```

Program Source Code:-

```
//Functions for matrix operations: addition, multiplication, substraction, transpose
#include<iostream>
#define SIZE 10
using namespace std;
//Function prototypes
void mat_add(int mat1[][SIZE], int mat2[][SIZE],int row1,int col1,int row2,int col2, int res[][SIZE]);
void mat_sub(int mat1[][SIZE], int mat2[][SIZE],int row1,int col1,int row2,int col2, int res[][SIZE]);
void mat_mul(int mat1[][SIZE], int mat2[][SIZE],int row1,int col1,int row2,int col2, int res[][SIZE]);
void mat_trans(int mat[][SIZE],int row,intcol,int res[][SIZE]);
int main(void)
{
        int A[SIZE][SIZE], B[SIZE][SIZE],result[SIZE][SIZE],
                row1,col1,row2,col2, choice;
        char ch = 'n';

        cout<<"Enter number of rows and columns of first matrix: ";
        cin>>row1>>col1;

        cout<<"Enter number of rows and columns of second matrix: ";
        cin>>row2>>col2;

        cout<<"Enter elements of first matrix:\n";
        for(int i=0; i<row1; i++)
                for(int j=0; j<col1; j++)
                        cin>>A[i][j];

        cout<<"\nEnter elements of second matrix:\n";
        for(int i=0; i<row2; i++)
                for(int j=0; j<col2; j++)
                        cin>>B[i][j];
        do
        {
                cout<<"1. Matrix addition\n"
                        <<"2. Matrix substraction\n"
```

```
                <<"3. Matrix multiplication\n"
                <<"4. Matrix transpose\n"
                <<"5. Exit\n";
        cin>>choice;
        switch(choice)
        {
                case 1:
                        mat_add(A,B,row1,col1,row2,col2,result);
                        break;
                case 2:
                        mat_sub(A,B,row1,col1,row2,col2,result);
                        break;
                case 3:
                        mat_mul(A,B,row1,col1,row2,col2,result);
                        break;
                case 4:
                        mat_trans(A,row1,col1,result);
                        mat_trans(B,row2,col2,result);
                        break;
                default:
                        return 0;
        }
        cout<<"\nDo you want to continue?(y/n) ";
        cin.sync();
        cin>>ch;
    }while(ch == 'Y' || ch=='y');
    return 0;
}

//Function to add 2 matrices
void mat_add(int mat1[][SIZE], int mat2[][SIZE],int row1,int col1,int row2,int col2, int res[][SIZE])
{
    if(row1 != row2 || col1 != col2)
    {
        cout<<"Addition not possible due to different dimensions\n";
        return;
    }

    for(int i=0; i<row1; i++)
        for(int j=0; j<col1;j++)
            res[i][j] = mat1[i][j] + mat2[i][j];
    //display
    cout<<"Result:\n";
    for(int i=0; i<row1; i++)
    {
        for(int j=0; j<col1;j++)
            cout<<res[i][j]<<" ";
        cout<<endl;
    }
}
//Function to subtract 2 matrices
```

```
void mat_sub(int mat1[][SIZE], int mat2[][SIZE],int row1,int col1,int row2,int col2, int res[][SIZE])
{
	if(row1 != row2 || col1 != col2)
	{
		cout<<"Subtraction not possible due to different dimensions\n";
		return;
	}
	for(int i=0; i<row1; i++)
		for(int j=0; j<col1;j++)
			res[i][j] = mat1[i][j] - mat2[i][j];
	//display
	cout<<"Result:\n";
	for(int i=0; i<row1; i++)
	{
		for(int j=0; j<col1;j++)
			cout<<res[i][j]<<" ";
		cout<<endl;
	}
}
//Function to multiply 2 matrices
void mat_mul(int mat1[][SIZE], int mat2[][SIZE],int row1,int col1,int row2,int col2, int res[][SIZE])
{
	if(col1 != row2)
	{
		cout<<"Multiplication not possible\n";
		return;
	}
	for(int i=0; i<row1; i++)
		for(int j=0; j<col2;j++)
			for(int k=0; k<col1; k++)
			res[i][j] += mat1[i][k] * mat2[k][j];
	//display
	cout<<"Result:\n";
	for(int i=0; i<row1; i++)
	{
		for(int j=0; j<col2;j++)
			cout<<res[i][j]<<" ";
		cout<<endl;
	}
}
//Function to transpose a matrix
void mat_trans(int mat[SIZE][SIZE],int row,intcol,int res[SIZE][SIZE])
{
	for(int i=0; i<row; i++)
		for(int j=0; j<col; j++)
			res[j][i] = mat[i][j];
	//display
	cout<<"\nResult:\n";
	for(int i=0; i<row; i++)
	{
		for(int j=0; j<col;j++)
```

```
                        cout<<res[i][j]<<endl;
                cout<<endl;
        }
}
```

Output:-

```
Enter number of rows and columns of first matrix: 2
2
Enter number of rows and columns of second matrix: 2
2
Enter elements of first matrix:
1
2
3
4
Enter elements of second matrix:
4
3
2
1
1. Matrix addition
2. Matrix substraction
3. Matrix multiplication
4. Matrix transpose
5. Exit
1
Result:
5 5
5 5
Do you want to continue?(y/n) y
1. Matrix addition
2. Matrix substraction
3. Matrix multiplication
4. Matrix transpose
5. Exit
2
Result:
-3 -1
1 3
Do you want to continue?(y/n) y
1. Matrix addition
2. Matrix substraction
3. Matrix multiplication
4. Matrix transpose
5. Exit
3
Result:
5 4
21 16
```

Do you want to continue?(y/n) y
1. Matrix addition
2. Matrix substraction
3. Matrix multiplication
4. Matrix transpose
5. Exit
4
Result:
13
24
Result:
4 2
31
Do you want to continue?(y/n) n

PRECAUTIONS AND DISCUSSION:

DISCUSSION:

1. Matrix is an arrangement of number, symbols, etc. in rows and columns , treated as a single quantity.
2. Conditions for matrix addition, subtraction, multiplication must be checked before performing operations.
3. In case of operator overloading, the object on the right hand side of the operator is always assumed as argument by compiler.

PRECAUTIONS:

1. Each line is ended with ";"(semicolon) to maintain the format rules.
2. The closing parameter of the class is followed by a ";"(Semicolon)
3. \N[mespace and proper header files are included to ensure proper execution of the library functions.
4. An infinite loop is used for proper execution according to the user.
5. Proper exit conditions are given to avoid incorrect outputs.
6. The array input for binary search must be in sorted order(for this program it has to be in ascending order).
7. If else condition is used for better execution of the program.

Assignment No :- 9

Program Statement:-

Design a template class to represent a generic vector. Provide member functions for the following:

a. Create a vector.
b. Addition of two vectors (Use overload opeartor)
c. Display the vector in the form
   {a,b,c,…….}

Program Algorithm:-

Description:

Vector containers are implemented as dynamic arrays; Just as regular arrays, vector containers have their elements stored in contiguous storage locations, which means that their elements can be accessed not only using iterators but also using offsets on regular pointers to elements. But unlike regular arrays, storage in vectors is handled automatically, allowing it to be expanded and contracted as needed.

Steps:

STEP 1: Define a generic class named vector as follows:
1.1 Define private data members size and a generic pointer vect
1.2 Define all member procedures as public
1.3 PROCEDUREvector()
1.3.1 Set size = 0
1.3.2 Set vect = NULL
End of Procedure
1.4 PROCEDURE vector(x)
1.4.1 Set size = x
1.4.2 Set vect = new array of generic type of size x
End of Procedure
1.5 PROCEDURE ~vector()
1.5.1 Delete dynamically allocated array of generic type pointed by vect
End of Procedure
1.6 PROCEDURE create()
1.6.1 Print “Enter elements: “
1.6.2 Set i = 0
1.6.3 Repeat until i>= size
Input vect[i]
End of Loop
End of Procedure
1.7 PROCEDURE display()
1.7.1 Print “(“
1.7.2 Set i = 0

1.7.3 Repeat until i>= size
Print vect[i]
End of Loop
1.7.4 Print “)”
End of Procedure
1.8 PROCEDURE operator +(constant object of class vector v)
1.8.1 Set i = 0
1.8.2 Repeat until i>= size
temp.vect[i] = v.vect[i] + vect[i]
End of Loop
1.8.3 Return object temp
End of Procedure
End of class definition
STEP 2: Print “Enter size of vectors: “
STEP 3: Input size
STEP 4: Declare objects of class vector of type integer: v1(size), v2(size), v3(size)
STEP 4: Call procedure v1.create()
STEP 5: Call procedure v2.create()
STEP 6: v3 = v1 + v2
STEP 7: Call procedure v1.display()
STEP 8: Call procedure v2.display()
STEP 9: Call procedure v1.display()
STEP 10: STOP

Program Source Code:-

```
// Header File
#include<iostream>

using namespace std;

template<class T>class Vector
{
	private:
		T *a;
		int n;
	public:
		Vector(int i=0)
		{
			n=i;
			a=new T[n];
		}

		void create()
		{
			int i;
			if(!n)
			{
				cout<<"\nEnter the Size of the Vector =";
				cin>>this->n;
```

```
                Vector(this->n);
            }
            for(i=0;i<n;i++)
            {
                cout<<"Enter A["<<i<<"] =";
                cin>>a[i];
            }
        }

        void display()
        {
            cout<<"{";
            for(int i=0;i<n;i++)
             cout<<" "<<a[i]<<" ";
            cout<<"}";
        }

        void add(Vector x,Vector b)
        {
            int i;
            if(x.n>=b.n)
            {
                this->n=x.n;
                Vector(this->n);
                for(i=0;i<b.n;i++)
                 a[i]=x.a[i]+b.a[i];
                for(;i<x.n;i++)
                 a[i]=x.a[i];
            }
            else
            {
                this->n=b.n;
                Vector(this->n);
                for(i=0;i<x.n;i++)
                 a[i]=x.a[i]+b.a[i];
                for(;i<b.n;i++)
                 a[i]=b.a[i];
            }
        }
};

main()
{
    Vector<float> v1(5),v2,v3;
    cout<<"Enter the Values for Vector V1 :-\n";
    v1.create();
    cout<<"\nVector V1: ";
    v1.display();
    cout<<"\n\nEnter the Values for Vector V2 :-\n";
    v2.create();
    cout<<"\nVector V2: ";
```

```
        v2.display();
        cout<<"\n\nAfter V1 + V2: ";
        v3.add(v1,v2);
        v3.display();
}
```

Output:-

SET 1:-

```
Enter the Values for Vector V1 :-
Enter A[0] =3
Enter A[1] =6
Enter A[2] =4
Enter A[3] =2
Enter A[4] =8

Vector V1: { 3  6  4  2  8 }

Enter the Values for Vector V2 :-

Enter the Size of the Vector =5
Enter A[0] =1
Enter A[1] =2
Enter A[2] =9
Enter A[3] =5
Enter A[4] =4

Vector V2: { 1  2  9  5  4 }

After V1 + V2: { 4  8  13  7  12 }
--------------------------------
Process exited after 12.85 seconds with return value 0
Press any key to continue . . .
```

SET 2:-

```
Enter the Values for Vector V1 :-
Enter A[0] =4
Enter A[1] =8
Enter A[2] =2
Enter A[3] =5
Enter A[4] =9

Vector V1: { 4  8  2  5  9 }

Enter the Values for Vector V2 :-

Enter the Size of the Vector =7
Enter A[0] =1
Enter A[1] =8
```

```
Enter A[2] =5
Enter A[3] =3
Enter A[4] =8
Enter A[5] =2
Enter A[6] =3

Vector V2: { 1  8  5  3  8  2  3 }

After V1 + V2: { 5  16  7  8  17  2  3 }
--------------------------------
Process exited after 14.24 seconds with return value 0
Press any key to continue . . .
```

PRECAUTIONS AND DISCUSSION:

DISCUSSION:

1. Vector is a geometric object that has magnitude and direction. Vectors can be added to another vector according to vector algebra. In this program , we add the two vectors using operator overload.
2. In case of operator overloading, the object on the right hand side of the operator is always assumed as argument by compiler.
3. In this program we have used a template class to accept all kinds of data instead of just normal integer.

PRECAUTIONS:

1. Each line is ended with ";"(semicolon) to maintain the format rules.
2. The closing parameter of the class is followed by a ";"(Semicolon)
3. Namespace and proper header files are included to ensure proper execution of the library functions.
4. An infinite loop is used for proper execution according to the user.
5. Proper exit conditions are given to avoid incorrect outputs.
6. The array input for binary search must be in sorted order(for this program it has to be in ascending order).
7. If else condition is used for better execution of the program.

Assignment No :- 10

Program Statement:-

Design a template class such that they support the following data conversions.

Dollar<float>dl,d2
Rupee<float>r1,r2.
d1=r1, r2=d2.

Program Algorithm:-

Description:

This is a program where two data types are there and we have to convert one data type to one data type value by this program. One data type name is Dollar and other one is Rupee.

Steps:

Step 1: creating the function **dollar(a=0)**
[this is the constructor of the dollar data type]

Step 1.1: d=a

Step 2: creating the function **set_data()**for dollar data type
[this is for setting the data]

Step 2.1: print “enter the data =”
read d
Step 2.2: STOP

Step 3: creating the function **display()**for dollar data type
[this is the function to print the data]

Step 3.1: print d
Step 3.2: STOP

Step 4: creating the function **overload the operator =(&a)** for dollar data type
[this is a function that overload the operator =. Here we passes the a argument of rupee data type.]

Step 4.1: d ← a.r*0.014
STOP

Step 5: creating the function **rupee(a = 0)**
[this is the constructor of the rupee data type]

Step 5.1: r ← a

Step 6: creating the function **set_data()** for rupee data type
[this is for setting the data]

Step 6.1: print “enter the data =”
read r
Step 6.2: STOP

Step 7: creating the function **display()**for rupee data type
[this is the function to print the data]

Step 7.1: print r
Step 7.2: STOP

Step 8: creating the function **overload the operator =(&a)** for rupee data type
[this is a function that overload the operator =. Here we passes the a argument of dollar data type.]

Step 8.1: r ← a.d*71.28
STOP

Step 9: creating the function **main()**

Step 9.1: d1 is the object of dollar data type
r1 is the object of rupee data type
Step 9.2: r1.set_data()
r1.display()
Step 9.3: d1=r1
print “After d = r ”
d1.display()
Step 9.4: d1.set_data()
d1.display()
Step 9.5: r1=d1
print “After r = d ”
r1.display()
Step 9.6: STOP

Program Source Code:-

```
// Header File
#include<iostream>
```

```
using namespace std;

// Classes and Prototypes
template<class>
class dollar;

template<class>
class rupee;

// Class dollar
template<class T1>
class dollar
{
	private:
		T1 d;
	public:
		dollar(T1 a=0)
		{
			d=a;
		}

		void set_data()
		{
			cout<<"Enter the value for Dollar =";
			cin>>d;
		}

		void display()
		{
			cout<<"Value ="<<d<<endl;
		}

		template<class>
		friend class rupee;

		template<class T2>
		void operator =(rupee<T2> &a)
		{
			d=a.r*0.014;
		}
};

// Class rupee
template<class T2>
class rupee
{
	private:
		T2 r;
	public:
		rupee(T2 a=0)
```

```
		{
			r=a;
		}

		void set_data()
		{
			cout<<"Enter the value for Dollar =";
			cin>>r;
		}

		void display()
		{
			cout<<"Value ="<<r<<endl;
		}

		template<class>
		friend class dollar;

		template<class T1>
		void operator =(dollar<T1> &a)
		{
			r=a.d*71.28;
		}
};


main()
{
	dollar<float> d1;
	rupee<float> r1;
	r1.set_data();
	cout<<"R: ";
	r1.display();
	d1=r1;
	cout<<"After  D = R :-\nD: ";
	d1.display();
	cout<<endl;
	d1.set_data();
	cout<<"D: ";
	d1.display();
	r1=d1;
	cout<<"After  R = D :-\nR: ";
	r1.display();
}
```

Output:-

SET 1:-

```
Enter the value for Rupee =100
```

R: Value =100
After D = R :-
D: Value =1.4

Enter the value for Dollar =10
D: Value =10
After R = D :-
R: Value =712.8

Process exited after 10.96 seconds with return value 0
Press any key to continue . . .

SET 2:-
Enter the value for Rupee =120
R: Value =120
After D = R :-
D: Value =1.68

Enter the value for Dollar =23
D: Value =23
After R = D :-
R: Value =1639.44

Process exited after 10.13 seconds with return value 0
Press any key to continue . . .

PRECAUTIONS AND DISCUSSION:

DISCUSSION:

1. A class created from class template is called template class. The syntax for definition of an object of a template class is
   Classname<type>objectname (arglist).
2. The array is initialized by the constructor. Here the member functions are defined inside the public visibility labels such that it is accessible by the object of the associated class declared inside the main function. The member function can wrap the private data from the outside function and we can call another member function from another member function. Every member function that is defined outside the template class follows that syntax….
   Returntypeclassname<T> :: function name(arglist)
   and it is preceded by this template<class T>.

PRECAUTIONS:

1. Each line is ended with ";"(semicolon) to maintain the format rules.
2. The closing parameter of the class is followed by a ";"(Semicolon)
3. Namespace and proper header files are included to ensure proper execution of the library functions.

4. An infinite loop is used for proper execution according to the user.
5. Proper exit conditions are given to avoid incorrect outputs.
6. The array input for binary search must be in sorted order(for this program it has to be in ascending order).
7. If else condition is used for better execution of the program.

Assignment No :- 11

Program Statement:-

Write an object oriented program using C ++ to create polynomial type object these polynomial type objects are created using singly Link List type object, which are again created using node type multi-field object , every object may take different type of numeric data values (use Template).Each object is created with their related operations ,In addition to do the followings for the polynomial type object:

a) Addition of two polynomials - using function and operator overloading .
b) Subtraction of two polynomials -- using function and operator overloading.
c) Multiplication of two polynomials -- using function and operator overloading.
d) Evaluation of a polynomials -- using Horner's rule
e) Creation of polynomials using input redirection operation(>>)
f) Printing the polynomials using output redirection (<<)

All kind of exception must be properly handled by the application system.

Program Algorithm:-

Description:

This is a program that implements polynomial function creation and displaying of polynomial functions. We can add two polynomial and multiply also by the functions, and we overload the operator to do that. Here we create a user defined data type named polynomial which have a coefficient part identified by c , a exponent part identified by e and a address part that can hold the address of the polynomial data type identified by next. The c, e, next are in the private part and the other functions are in the public section.

Steps:

Step 1: creating the **polynomial(a ← 0,b ← 0)**
[This the constructor it takes two argument a and b which are passes at the time of creating the object, if it not passes then both a, b are assigned to 0]

Step 1.1: c ← a
e ← b
next ← NULL

Step 2: creating the function for **overload the operator >>(&in,&h)**
[This function is use to create a polynomial function where we the coefficient and the exponent from the user. this function take two reference type argument one from istream type object type names in and another is polynomial type object name h]

Step 2.1: p and q are two pointers of polynomial data types
a ← 'y'
Step 2.2: Loop continue when a = 'Y' or a = 'y' true
Step 2.2.1: print "Enter the Coefficient = "
read c
print "Enter the exponent ="
read e
Step 2.2.2: create a object of polynomial data type by polynomial(c,e) and return it's address to the p
Step 2.2.3: if h.next = NULL then
h.next ← p
else
next[q] ← p
[end of if]
Step 2.2.4: q ← p
print "Do you want to continue (y/n) ="
read a
[end of Loop]
Step 2.3: STOP

Step 3: creating the function **overload operator <<(&out,&h)**
[This is a function that display the polynomial function into the monitor. this function take two reference type argument one from ostream type object type names out and another is polynomial type object name h]

Step 3.1: create a pointer p of polynomial data type
p ← h.next
Step 3.2: if p = NULL then
print "No Polynomial function exists."
STOP
[end of if]
Step 3.3: if c[p] > 0 then
print c[p] , "X^", e[p], " "
else
print -1*c[p], "X^", e[p], " "
[end of if]
Step 3.4: Loop continue when p != NULL true
if c[p] > 0 then
print c[p] , "X^", e[p], " "
else
print -1*c[p], "X^", e[p], " "
[end of if]
p ← next[p]
[End of Loop]
Step 3.5: STOP

Step 4: creating the function **overload the operator +(b)**

[This is a function that takes a argument and add it to the calling object and store it into a object and return it]

Step 4.1: p, q, r, t are pointers of polynomial data type
h is a object of the polynomial
p ← next[this]
q ← b.next
Step 4.2: Loop continue when p != NULL and q != NULL true
Step 4.2.1: if e[p] = e[q] then
r ← a address of a new object by polynomial(c[p]+c[q],e[p])
p ← next[p]
q ← next[q]
Step 4.2.2: else
if e[p] > e[q] then
r ← a address of a new object by polynomial(c[p], e[p])
p ← next[p]
else
r ← a address of a new object by polynomial(c[q], e[q])
q ← next[q]
[end of if]
[end of if]
Step 4.2.3: if h.next = NULL then
h.next ← r
else
next[t] ← r
[end of if]
Step 4.2.4: t ← r
[end of Loop]
Step 4.3: Loop continue when p!= NULL true
Step 4.3.1: r ← a address of a new object by polynomial(c[p], e[p])
p ← next[p]
Step 4.3.2: if h.next = NULL then
h.next ← r
else
next[t] ← r
[end of if]
Step 4.3.3: t ← r
[End of Loop]
Step 4.4: Loop continue when q!= NULL true
Step 4.4.1: r ← a address of a new object by polynomial(c[q], e[q])
q ← next[q]
Step 4.4.2: if h.next = NULL then
h.next ← r
else
next[t] ← r
[end of if]
Step 4.4.3: t ← r
[End of Loop]
Step 4.4: return h
STOP

Step 5: creating the function **overload the operator *(b)**
[This is a function that takes a argument and multiply it to the calling object and store it into a object and return it]

Step 5.1: q, p, r, t are the pointers of the polynomial data type
h1, h2 are the objects of polynomial data type
p ← next[this]
Step 5.2: Loop continue when p != NULL true
Step 5.2.1: h1.next ← NULL
q ← b.next
Step 5.2.2: Loop continue when q!= NULL true
r ← address of a new object by polynomial(c[p]*c[q],e[p]*e[q])
if h1.next = NULL then
h1.next ← r
else
next[t] ← r

[end of if]
t ← r
q ← next[q]
[end of Loop]
Step 5.2.3: h2= h1+h2
p ← next[p]
[end of Loop]
Step 5.3: return h2
STOP

Step 6: creating the function **overload the operator -(b)**
[This is a function that takes a argument and subtract it to the calling object and store it into a object and return it]

Step 6.1: p, q, r, t are pointers of polynomial data type
h is a object of the polynomial
p ← next[this]
q ← b.next
Step 6.2: Loop continue when p != NULL and q != NULL true
Step 6.2.1: if e[p] = e[q] then
r ← a address of a new object by polynomial(c[p]-c[q],e[p])
p ← next[p]
q ← next[q]
Step 6.2.2: else
if e[p] > e[q] then
r ← a address of a new object by polynomial(c[p], e[p])
p ← next[p]
else
r ← a address of a new object by polynomial(-c[q], e[q])
q ← next[q]
[end of if]
[end of if]

Step 6.2.3: if h.next = NULL then
h.next ← r
else
next[t] ← r
[end of if]
Step 6.2.4: t ← r
[end of Loop]
Step 6.3: Loop continue when p!= NULL true
Step 6.3.1: r ← a address of a new object by polynomial(c[p], e[p])
p ← next[p]
Step 6.3.2: if h.next = NULL then
h.next ← r
else
next[t] ← r
[end of if]
Step 6.3.3: t ← r
[End of Loop]
Step 6.4: Loop continue when q!= NULL true
Step 6.4.1: r ← a address of a new object by polynomial(-c[q], e[q])
q ← next[q]
Step 6.4.2: if h.next = NULL then
h.next ← r
else
next[t] ← r
[end of if]
Step 6.4.3: t ← r
[End of Loop]
Step 6.4: return h
STOP

Step 7: creating the function **evaluation(x)**
[This is function takes a value and evaluate the value of the polynomial function.]

Step 7.1: p is the pointer of the polynomial data type
sum← 0
p ← next[this]
Step 7.2: Loop continue when p!=NULL true
sum ← sum + c[p]*x^e[p]
p ← next[p]
[end of Loop]
Step 7.3: return sum
STOP

Step 8: creating the **main()**

Step 8.1: h1, h2 , h3 are objects of the polynomial data type
Step 8.2: read h1
print h1
Step 8.3: read h2

print h2

Step 8.4: h3=h1+h2
print h3

Step 8.5: h3=h1-h2
print h3

Step 8.6: h3=h1*h2
print h3

Step 8.7: print “Enter the value for evaluation =”
read x
print “first polynomial value =” , h1.evaluation(x)
print “Second polynomial value =”, h2.evaluation(x)

Program Source Code:-

```
// Header Files
#include<iostream>
#include<cmath>

using namespace std;

template<class T>
class polynomial
{
	private:
		T c;
		int e;
		polynomial *next;
	public:
		polynomial(T a=0,int b=0)
		{
			c=a;
			e=b;
			next=NULL;
		}

		friend void operator >>(istream &in,polynomial &b)
		{
			polynomial *p,*q;
			T c;
			int e;
			char a='y';
			while(a=='Y'||a=='y')
			{
				cout<<"\nEnter the cofficent =";
				cin>>c;
				cout<<"Enter the exponent =";
				cin>>e;
				p=new polynomial(c,e);
				if(b.next==NULL)
```

```
                b.next=p;
               else
                q->next=p;
               q=p;
               cout<<"Do you want to continue (y/n) =";
               cin>>a;
        }
}

friend void operator <<(ostream &a,polynomial b)
{
        polynomial *p;
        p=b.next;
        if(p==NULL)
        {
                cout<<"No Polynomial Exsits.";
                return;
        }
        if(p->c>0)
         cout<<p->c<<"X^"<<p->e<<" ";
        else
         cout<<"- "<<-1*p->c<<"X^"<<p->e<<" ";
        p=p->next;
        while(p!=NULL)
        {
                if(p->c>0)
                 cout<<"+ "<<p->c<<"X^"<<p->e<<" ";
                else
                 cout<<"- "<<-1*p->c<<"X^"<<p->e<<" ";
                p=p->next;
        }
}

polynomial operator +(polynomial b)
{
        polynomial *p,*q,*r,*t,h;
        p=this->next;
        q=b.next;
        while(p!=NULL and q!=NULL)
        {
                if(p->e==q->e)
                {
                        r=new polynomial(p->c+q->c,p->e);
                        p=p->next;
                        q=q->next;
                }
                else
                 if(p->e>q->e)
                 {
                        r=new polynomial(p->c,p->e);
                        p=p->next;
```

```
                }
                else
                {
                        r=new polynomial(q->c,q->e);
                        q=q->next;
                }
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
        }
        while(p!=NULL)
        {
                r=new polynomial(p->c,p->e);
                p=p->next;
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
        }
        while(q!=NULL)
        {
                r=new polynomial(q->c,q->e);
                q=q->next;
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
        }
        return h;
}

polynomial operator -(polynomial b)
{
        polynomial *p,*q,*r,*t,h;
        p=this->next;
        q=b.next;
        while(p!=NULL and q!=NULL)
        {
                if(p->e==q->e)
                {
                        r=new polynomial(p->c-q->c,p->e);
                        p=p->next;
                        q=q->next;
                }
                else
                 if(p->e>q->e)
                 {
```

```
                        r=new polynomial(p->c,p->e);
                        p=p->next;
                }
                else
                {
                        r=new polynomial(-q->c,q->e);
                        q=q->next;
                }
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
        }
        while(p!=NULL)
        {
                r=new polynomial(p->c,p->e);
                p=p->next;
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
        }
        while(q!=NULL)
        {
                r=new polynomial(-q->c,q->e);
                q=q->next;
                if(h.next==NULL)
                 h.next=r;
                else
                 t->next=r;
                t=r;
        }
        return h;
}

polynomial operator *(polynomial b)
{
        polynomial h1,h2,*p,*q,*r,*t;
        p=this->next;
        while(p!=NULL)
        {
                h1.next=NULL;
                q=b.next;
                while(q!=NULL)
                {
                        r=new polynomial(p->c*q->c,p->e*q->e);
                        if(h1.next==NULL)
                         h1.next=r;
                        else
```

```
                                        t->next=r;
                                        t=r;
                                        q=q->next;
                                }
                                h2=h1+h2;
                                p=p->next;
                        }
                        return h2;
                }

                float evaluation(float x)
                {
                        polynomial *p;
                        float sum=0;
                        p=this->next;
                        while(p!=NULL)
                        {
                                sum=sum+p->c*(pow(x,p->e));
                                p=p->next;
                        }
                        return sum;
                }
};


main()
{
        polynomial<int> h1,h2,h3;
        int x;
        cout<<"Enter the First Polynomial :-";
        cin>>h1;
        cout<<"\nFirst Polynomial :-  ";
        cout<<h1;
        cout<<"\n\nEnter the Second Polynomial :-";
        cin>>h2;
        cout<<"\nSecond Polynomial :-  ";
        cout<<h2;
        h3=h1+h2;
        cout<<"\n\n\nAfter H1 + H2 :-   ";
        cout<<h3;
        h3=h1-h2;
        cout<<"\n\n\nAfter H1 - H2 :-   ";
        cout<<h3;
        h3=h1*h2;
        cout<<"\n\nAfter H1 * H2 :-   ";
        cout<<h3;
        cout<<"\n\nEnter the value of x to Evaluate =";
        cin>>x;
        cout<<"\nFirst Polynomial :-  ";
        cout<<h1;
        cout<<"\nValue of it ="<<h1.evaluation(x);
```

```
        cout<<"\n\nSecond Polynomial :-  ";
        cout<<h2;
        cout<<"\nValue of it ="<<h1.evaluation(x);
}
```

Output:-

SET 1:-

```
Enter the First Polynomial :-
Enter the cofficent =5
Enter the exponent =6
Do you want to continue (y/n) =y

Enter the cofficent =4
Enter the exponent =5
Do you want to continue (y/n) =y

Enter the cofficent =8
Enter the exponent =2
Do you want to continue (y/n) =n

First Polynomial :-  5X^6 + 4X^5 + 8X^2

Enter the Second Polynomial :-
Enter the cofficent =2
Enter the exponent =7
Do you want to continue (y/n) =y

Enter the cofficent =4
Enter the exponent =5
Do you want to continue (y/n) =y

Enter the cofficent =4
Enter the exponent =3
Do you want to continue (y/n) =n

Second Polynomial :-  2X^7 + 4X^5 + 4X^3


After H1 + H2 :-   2X^7 + 5X^6 + 8X^5 + 4X^3 + 8X^2


After H1 - H2 :-   - 2X^7 + 5X^6 - 0X^5 - 4X^3 + 8X^2

After H1 * H2 :-   10X^42 + 8X^35 + 20X^30 + 16X^25 + 20X^18 + 16X^15 + 16X^14 + 32X^10
+ 32X^6

Enter the value of x to Evaluate =2
```

First Polynomial :- 5X^6 + 4X^5 + 8X^2
Value of it =480

Second Polynomial :- 2X^7 + 4X^5 + 4X^3
Value of it =480

Process exited after 68.18 seconds with return value 0
Press any key to continue . . .

SET 2:-
Enter the First Polynomial :-
Enter the cofficent =2
Enter the exponent =4
Do you want to continue (y/n) =y

Enter the cofficent =3
Enter the exponent =2
Do you want to continue (y/n) =n

First Polynomial :- 2X^4 + 3X^2

Enter the Second Polynomial :-
Enter the cofficent =6
Enter the exponent =3
Do you want to continue (y/n) =y

Enter the cofficent =4
Enter the exponent =3
Do you want to continue (y/n) =n

Second Polynomial :- 6X^3 + 4X^3

After H1 + H2 :-   2X^4 + 6X^3 + 4X^3 + 3X^2

After H1 - H2 :-   2X^4 - 6X^3 - 4X^3 + 3X^2

After H1 * H2 :-   12X^12 + 8X^12 + 18X^6 + 12X^6

Enter the value of x to Evaluate =9

First Polynomial :- 2X^4 + 3X^2
Value of it =13365

Second Polynomial :- 6X^3 + 4X^3
Value of it =13365

Process exited after 110.7 seconds with return value 0
Press any key to continue . . .

PRECAUTIONS AND DISCUSSION:

DISCUSSION:

1. This program uses template type class.Template can be used as macro which is an approach to generic programming.Since a template can be defined with a parameter thus it can be replaced by a specific data type at time of actual use of function.Class poly and add2poly are declared as template type.

2. Block 'polynomial' is declared comprising of two data parts c and e and node pointer to track the polynomial. e is the degree of the polynomial whereas c is the coefficient of the polynomial.

3. Here, the operators '+', '-' and '*' are overloaded to perform Polynomial ADDITION, SUBTRACTION and MULTIPLICATION respectively.

PRECAUTIONS:

1. Each line is ended with ";"(semicolon) to maintain the format rules.
2. The closing parameter of the class is followed by a ";"(Semicolon)
3. Namespace and proper header files are included to ensure proper execution of the library functions.
4. An infinite loop is used for proper execution according to the user.
5. Proper exit conditions are given to avoid incorrect outputs.
6. The array input for binary search must be in sorted order(for this program it has to be in ascending order).
7. If else condition is used for better execution of the program.