KARPAGAM ACADEMY OF HIGHER EDUCATION

KARPAGAM UNIVERSITY

(Deemed University Under Section 3 of UGC Act 1956)

Coimbatore -21.

DEPARTEMENT OF COMPUTER SCIENCE

SYLLABUS

Semester – II

16CSU311 DATA STRUCTURES - PRACTICAL 4H – 2C

Instruction Hours / week: L: 0 T: 0 P: 4 Marks: Int : **40** Ext : **60** Total: **100**

1. Write a program to search an element from a list. Give user the option to perform Linear or Binary search. Use Template functions.
2. WAP using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3. Implement Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists (include a function and also overload operator +).
4. Implement Doubly Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
5. Implement Circular Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
6. Perform Stack operations using Linked List implementation.
7. Perform Stack operations using Array implementation. Use Templates.
8. Perform Queues operations using Circular Array implementation. Use Templates.
9. Create and perform different operations on Double-ended Queues using Linked List implementation.
10. WAP to scan a polynomial using linked list and add two polynomial.
11. WAP to calculate factorial and to compute the factors of a given no. (i)using recursion, (ii) using iteration
12. (ii) WAP to display fibonaCSUi series (i)using recursion, (ii) using iteration
13. WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion
14. WAP to create a Binary Search Tree and include following operations in tree: (a) Insertion (Recursive and Iterative Implementation)
(b) Deletion by copying
(c) Deletion by Merging
(d) Search a no. in BST
(e) Display its preorder, postorder and inorder traversals Recursively
(f) Display its preorder, postorder and inorder traversals Iteratively
(g) Display its level-by-level traversals
(h) Count the non-leaf nodes and leaf nodes
(i) Display height of tree
(j) Create a mirror image of tree
(k) Check whether two BSTs are equal or not
15. WAP to convert the Sparse Matrix into non-zero form and vice-versa.
16. WAP to reverse the order of the elements in the stack using additional stack.

17. WAP to reverse the order of the elements in the stack using additional Queue.
18. WAP to implement Diagonal Matrix using one-dimensional array.
19. WAP to implement Lower Triangular Matrix using one-dimensional array.
20. WAP to implement Upper Triangular Matrix using one-dimensional array.
21. WAP to implement Symmetric Matrix using one-dimensional array.
22. WAP to create a Threaded Binary Tree as per inorder traversal, and implement operations like finding the suCSUessor / predecessor of an element, insert an element, inorder traversal.
23. WAP to implement various operations on AVL Tree.

Ex No: 01 **FACTORIAL AND FACTORS**

Date:

Aim:

To write a program for to calculate factorial and to compute the factors of a given no

i. Using Recursion
ii. Using Iteration

Algorithm for Factorial:

Step 1: Start the process
Step 2: Include the necessary header files and namespace.
Step 3: Declare the variables i,n,fact,f as integers
Step 4: Create a recursive function recur_factorial()
Step 5: In the main function, get the number n as input
Step 6: Calculate the factorial using a for loop
Step 7: Display the result
Step 8: Calculate the factorial by calling the function recur_factoiral()
Step 9: Display the result
Step 10: Stop the process**.**

Algorithm for Factors:

Step 1: Start the process
Step 2: Include the necessary header files and namespace.
Step 3: Declare the variables i,n as integers and read n.
Step 4: In main function calculate the factors of n inside a for loop
Step 5: Display the result
Step 6: Stop the process

Program Coding:

Factorial:

```
#include<iostream>
using namespace std;
int recur_factorial(int n)
{  return n>1?n*recur_factorial(n-1):1;
}
int main()
{ int i,n,fact,f;
f=1;
cout<<"\nenter the num: ";
cin>>n;
cout<<"\nCalculation using Iteration";
for(i=1; i<=n; ++i)
{f=f*i;}
cout<<"\nFactorial of " << n <<" is " << f<<"\n";
cout<<"Calculation using Recursion";
fact=recur_factorial(n);
cout<<"\nFactorial of " << n <<" is " << fact<<"\n";
return 0;
}
```

Output:

```
enter the num: 5
Calculation using Iteration
Factorial of 5 is 120
Calculation using Recursion
Factorial of 5 is 120
```

Factors:

```
#include<iostream>
using namespace std;
int main()
{int n,i;
cout<<"Enter the number to find its factors:"<<endl;
cin>>n;
cout<<"The factors of "<<n<<" are:"<<endl;
for(i=1;i<=n;i++)
{if(n%i==0)
cout<<"\t"<<i;
}
cout<<endl;
return 0;
}
```

Output:

```
Enter the number to find its factors:
6
The factors of 6 are:
1       2   3   6
```

Result:

The above program has been executed successfully and output is verified.

Ex No: 02 **FIBONACCI SERIES**

Date:

Aim:

To write a program to display the Fibonacci series using recursion and iteration.

Algorithm:

Step 1: Start the process

Step 2: Include the necessary header files and namespace.

Step 3: Declare the variables i,n,f,f1,f2 and c as integers

Step 4: Create a recursive function Fibonacci()

Step 5: In main function, get the number n as input and initialize i,f1 to 0 and f2 to 1

Step 6: Calculate the Fibonacci series using a for loop with f=f1+f2; f1=f2; f2=f;

Step 7: Display the result

Step 8: Calculate the Fibonacci series by calling the function Fibonacci(n)

Step 9: Display the result

Step 10: Stop the process.

Program Coding:

```
#include<iostream>
using namespace std;
int Fibonacci(int n)
{   if ( n == 0 )
      return 0;
   else if ( n == 1 )      return 1;
```

```
   else      return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
int main()
{  int n, i = 0, c;
  int f,f1=0,f2=1;
   cout<<"\nenter the no. of terms: ";
   cin>>n;
cout<<"FINONACCI SERIES USING RECURSION\t";
    for ( c = 1 ; c <= n ; c++ )
   {   cout<<Fibonacci(i) << "\n";
      i++;    }
cout<<"FINONACCI SERIES USING ITERATION\t";
cout<< f1 <<"\n" << f2;
 for(i=1;i<n-1;i++)
         {f=f1+f2;
         f1=f2;
         f2=f;
cout<<"\n"<<f;
}
    return 0;
}
```

Output :

```
enter the no. of terms: 6
FINONACCI SERIES USING RECURSION
0       1       1       2       3       5
FINONACCI SERIES USING ITERATION
0       1       1       2       3       5
```

Result:

The above program has been executed successfully and output is verified.

Ex No:3 **GCD OF TWO NUMBERS**

Date:

Aim:

To write a program to calculate GCD of 2 number (i) with recursion (ii) without recursion

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: Declare the variables num1,num2,result,t,a,b as int

STEP 4: Get the values for num1,num2

STEP 5: To calculate the gcd using recursion call gcd(num1,num2)

STEP 6: Display the result

STEP 7: Calculate the gcd of num1 and num2 within a while loop

STEP 8: Display the results

STEP 9: Stop the process

Program coding:

```
#include<iostream>
using namespace std;
int gcd(int u, int v)
{    return (v != 0) ? gcd(v, u % v) : u; }
 int main(void)
{   int num1, num2, result,t,a,b;
    int x,y;
   cout << "Enter two numbers to find GCD using recursion: ";
   cin >> num1 >> num2;
   result = gcd(num1, num2);
```

```
    if (gcd)
  cout << "\nThe GCD of " << num1 << " and " << num2 << " is: " <<
result     << endl;
    else
        cout << "\nInvalid input!!!\n";
cout << "Enter two numbers to find GCD  without using recursion: \n";
 cin >> a >> b;
x=a;
y=b;
while(y!=0)
 { t=y;
 y=x%y;
 x=t;
 }
  cout << "\nThe GCD of " << a << " and " << b << " is: " << x << endl;
  return 0;
}
```

Output:

```
Enter two numbers to find GCD using recursion:
4
8
The GCD of 4 and 8 is: 4
Enter two numbers to find GCD  without using recursion:
15
18
The GCD of 15 and 18 is: 3
```

Result:

The above program has been executed successfully and output is verified.

Ex No:4 **SYMMETRIC MATRIX**

Date:

Aim:

To write a program to implement Symmetric Matrix.

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: Declare variables r1, c1, r, c, size, isSymmetric as int

STEP 4: Declare two integer arrays inMat and transMat with size 50*50

STEP 5: Get the size of the square matrix. r1 = c1 = size;

STEP 6: Get the matrix elements inside a nested for loop

STEP 7: Transpose the matrix: transMat[c][r] = inMat[r][c];

STEP 8: Display both the matrices

STEP 9: Check whether the given matrix is same as the transposed matrix

STEP 10: If it's same then print symmetric matrix

STEP 11: Or else print not symmetric matrix

STEP 12: Stop the process

Program coding:

```
#include<iostream>
using namespace std;
int main()
{ int r1, c1, r, c, size, isSymmetric;
 int inMat[50][50], transMat[50][50];
 cout<<"Enter the size of Square Matrix\n";
  cin>>size;
  r1 = c1 = size;
  cout<<"Enter Matrix of size "<<r1 <<" X "<< c1<<endl;
```

```
        for(r = 0; r < r1; r++){
        for(c = 0; c < c1; c++){
           cin>>inMat[r][c];
        }   }
     /* Find Transpose of input  transpose[i][j] = input[j][i] */
    for(r = 0; r < r1; r++){
        for(c = 0; c < c1; c++){
        transMat[c][r] = inMat[r][c];
        }   }
     cout<<"The given matrix is"<<endl;
for(r = 0; r < r1; r++){
  for(c = 0; c < c1; c++){
   cout<<inMat[r][c]<<"  ";       }
cout<<endl;    }
cout<<"The transpose of the given matrix is"<<endl;
for(r = 0; r < r1; r++){
   for(c = 0; c < c1; c++){
    cout<<transMat[r][c]<<"  ";       }
cout<<endl; }
 isSymmetric = 1;
 for(r = 0; r < c1; r++){
for(c = 0; c < r1; c++){
if(inMat[r][c] != transMat[r][c])
{isSymmetric = 0; }       }    }
if(isSymmetric == 1)
  cout<<"Input Matrix is Symmetric Matrix\n";
  else
 cout<<"Input Matrix is Not a Symmetric Matrix\n";
  return 0; }
```

Output:

```
Enter the size of Square Matrix
2
Enter Matrix of size 2 X 2
1 2
2 1
The given matrix is
1  2
2  1
The transpose of the given matrix is
1  2
2  1
Input Matrix is Symmetric Matrix
Enter the size of Square Matrix
2
Enter Matrix of size 2 X 2
3 3
4 4
The given matrix is
3  3
4  4
The transpose of the given matrix is
3  4
3  4
Input Matrix is Not a Symmetric Matrix
```

Result:

The above program has been executed successfully and output is verified.

Ex No:5 STACK OPERATIONS USING ARRAY

Date:

Aim:

To write a program to Perform Stack operations using Array implementation using Templates.

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: Create a template class type

STEP 4: Create a class stack and declare an array of size 10

STEP 5: Inside the constructor initialize the pointer top=-1

STEP 6: Read the size of the stack

STEP 7: Create the functions push, pop

STEP 8: Using Switch case statement select one appropriate operation

1. PUSH:
    i. Read an element
    ii. Store it at the top of the stack
    iii. Increment the top
2. POP
    i. Read the element from the top of the stack
    ii. Display it
    iii. Decrement the top

STEP 9: Stop the process

Program Coding:

```
#include<iostream>
using namespace std;
template<class Type>
```

```
class Stack
{Type s[10];
int top,n;
public:
Stack()
{ top=-1;
cout<<"\n\tEnter the Stack Size : ";
cin>>n;
}
void push(Type no)
{ if(top<n-1)
s[++top]=no;
else
cout<<"\n\tstack is full.Can't insert "<<no<<endl;
}
void pop()
{ if(top<0)
cout<<"\n\tstack is empty.\n";
else
cout<<"\n\tPoped  : "<<s[top--];
}
void stk_operation();
};
template<class Type>
void Stack<Type> :: stk_operation()
{ int choice=1,i;
Type no;
while(choice>0 && choice<3)
{ cout<<"\n\n\t1.PUSH\t2.POP\tAny Key To Exit\tChoice : ";
```

```
cin>>choice;
switch(choice)
{ case 1 : //push
cout<<"\n\tEnter the No to push : ";
cin>>no;
push(no);
cout<<"\n\t\tstack content :\n\n\t";
for(i=0;i<=top;i++)
cout<<s[i]<<"\t";
break;
case 2 : //pop
pop();
cout<<"\n\t\tstack content :\t";
for(i=0;i<=top;i++)
cout<<s[i]<<"\t";
break;
} } }
int main()
{ cout<<"\n\t\tSTACK OPERATION USING TEMPLATE\n\n";
Stack<int> stk1;
while(1)
{ stk1.stk_operation();
return(1);
} }
```

Output:

```
STACK OPERATION USING TEMPLATE
        Enter the Stack Size : 3
                1.PUSH    2.POP    Any Key To Exit   Choice : 1
```

```
Enter the No to push : 1
        stack content :     1
1.PUSH      2.POP       Any Key To Exit   Choice : 1
Enter the No to push : 2
        stack content :     1       2
1.PUSH      2.POP       Any Key To Exit   Choice : 1
Enter the No to push : 3
        stack content :     1       2       3
1.PUSH      2.POP       Any Key To Exit   Choice : 1
Enter the No to push : 4
stack is full.Can't insert 4
        stack content :     1       2       3
1.PUSH      2.POP       Any Key To Exit   Choice : 2
Poped  : 3
        stack content :     1       2
1.PUSH      2.POP       Any Key To Exit   Choice : 2
Poped  : 2
        stack content :     1
1.PUSH      2.POP       Any Key To Exit   Choice : 2
Poped  : 1
        stack content :
1.PUSH      2.POP       Any Key To Exit   Choice : 2
stack is empty.
1.PUSH      2.POP       Any Key To Exit   Choice : 3
```

Result:

The above program has been executed successfully and output is verified.

Ex No:6 QUEUE OPERATIONS USING ARRAY

Date:

Aim:

To write a program to perform queues operations using circular array implementation using Templates.

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: Define the max as 3, initialize front=0, rear=-1

STEP 4: Read ch and n value

STEP 5: check (ch<=4) and if (rear =n) then read value

STEP 6: Assign value to q(rear) and increment the rear, else Display as queue if full

STEP 7: if (front=rear) then print front value of queue and increment front

STEP 8: Else Display as queue is empty

STEP 9: if (rear=0) then assign front to I and write q[i].

STEP 10: Else Display as queue is empty

STEP 11: Go to step 5

STEP 12: Stop the process

Program Coding:

```
#include<iostream>
using namespace std;
#define max 3
int q[10],front=0,rear=-1;
int main()
{  int ch;
```

```
    void insert();
    void delet();
    void display();
    cout<<"\nCircular Queue operations\n";
    cout<<"1.insert\n2.delete\n3.display\n4.exit\n";
    while(ch<<=4)
    {  cout<<"Enter your choice:";
       cin>>ch;
       switch(ch)
       {   case 1: insert();
           continue;
       case 2: delet();
           continue;
       case 3:display();
           continue;
        case 4:break;
             }
break;
   }}
 void insert()
{  int x;
   if((front==0&&rear==max-1)||(front>0&&rear==front-1))
       cout<<"Queue is overflow\n";
   else    {  cout<<"Enter element to be insert:";
       cin>>x;
       if(rear==max-1&&front>0)
       {  rear=0;
          q[rear]=x;       }
       else       {    if((front==0&&rear==-1)||(rear!=front-1))
```

```
            q[++rear]=x;
        }   }}
void  delet()
{   int a;
   if((front==0)&&(rear==-1))
   {     cout<<"Queue is underflow\n";          }
else
{   if(front==rear)
   {  a=q[front];
      rear=-1;
      front=0;   }
   else
      if(front==max-1)
      {  a=q[front];
         front=0;       }
      else a=q[front++];
      cout<<"Deleted element is:"<<a<<endl;
}}
void display()
{   int i,j;
   if(front==0&&rear==-1)
   {       cout<<"Queue is underflow\n";       }
else{
   if(front>rear)
   {       for(i=0;i<=rear;i++)
           cout<<q[i];
      for(j=front;j<=max-1;j++)
           cout<<q[j];
cout<<"front="<<front<<" rear="<<rear<<endl;
```

```
        cout<<"\nrear is at "<<q[rear]<<endl;
        cout<<"\nfront is at "<<q[front]<<endl;
    }
    else
    {       for(i=front;i<=rear;i++)
        {           cout<<q[i]<<endl;       }
        cout<<"\nrear is at "<<q[rear];
        cout<<"\nfront is at "<<q[front];
    }
    cout<<"\n";
}}
```

Output:

```
Circular Queue operations
1.insert
2.delete
3.display
4.exit
Enter your choice:1
Enter element to be insert:1
Enter your choice:1
Enter element to be insert:2
Enter your choice:1
Enter element to be insert:3
Enter your choice:1
Queue is overflow
Enter your choice:3
1
2
```

3

rear is at 3

front is at 1

Enter your choice:2

Deleted element is:1

Enter your choice:2

Deleted element is:2

Enter your choice:2

Deleted element is:3

Enter your choice:2

Queue is underflow

Enter your choice:3

Queue is underflow

Enter your choice:4

Result:

The above program has been executed successfully and output is verified.

Ex No:7 SINGLY LINKED LIST OERATIONS

Date:

Aim:

To write a program to perform Implement Linked List using templates. Include functions for insertion, deletion and search of a number.

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: Create class node and a template

STEP 4: Create a class list. Initialize the node object first to NULL in the contructor

STEP 5: Create a function create_beg() to create an initial list

STEP 6: The function insert() inserts a node at a specific position.

STEP 7: Search method finds a node and the node can either be deleted or replaced.

STEP 8: The linked is displayed after each operation using display() function

STEP 9: Stop the process

Program Coding:

```
#include<iostream>
using namespace std;
template<class t>
class node
{ public:
t data;
node<t> *next;
```

```
};
template<class t1>
class list
{ int n;
node<int> *first;
public:
list()
{first=NULL; }
void create_beg()
{ cout<<"Enter how many nodes u want to enter in linked list:\n";
cin>>n;
node<int> *temp;
first=new node<int>;
cout<<"Enter first node data:\n";
cin>>first->data;
first->next=NULL;
for(int i=1;i<n;i++)
{ temp=new node<int>;
cout<<"Enter node data:\n";
cin>>temp->data;
temp->next=first;
first=temp;
} }
int count()
{node<int> *current;
int c=0;
current=first;
while(current!=NULL)
{c++;
```

```
current=current->next;
}
return c;}
void insert(t1 n)
{int b=count();
if(n<=b+1)
{node<int> *current,*temp;
current=first;
temp=new node<int>;
cout<<"Enter data:\n";
cin>>temp->data;
temp->next=NULL;
if(n==1)
{ temp->next=first;
first=temp;
}
else
{for(int i=1;i<n-1;i++)
current=current->next;
temp->next=current->next;
current->next=temp; }}
else
cout<<"Can't be inserted\n"; }
void search()
{int flag=0;
cout<<"Enter data to be searched:\n";
cin>>n;
node<int> *current,*prev,*temp;
int b=count();
```

```
current=first;
for(int i=1;i<=b;i++)
{if(current->data==n)
{ flag=1;
break; }
prev=current;
current=current->next; }
if(flag==1)
{int c;
cout<<"Data found:\nEnter your choice:\n1.delete data\n 2.replace data\n
3.do nothing\n";
cin>>c;
switch(c)
{case 1:temp=current;
prev->next=current->next;
delete(temp);
cout<<"Data deleted:\n";
break;
case 2:cout<<"Enter new data:\n";
cin>>current->data;
cout<<"data replaced:\n";
break;
case 3:break;
default:cout<<"wrong choice:\n"; }}
else
cout<<"Data not found:\n"; }
void display()
{node<int> *current;
current=first;
```

```
cout<<"The data in linked list:\n";
while(current!=NULL)
{cout<<current->data<<" -> ";
current=current->next;
}
cout<<endl; }};
int main()
{int cho,n;
char ch;
list<int> l1;
l1.create_beg();
l1.display();
do
{cout<<"Want to insert a node:\n";
cin>>ch;
if(ch=='y')
{ cout<<"Enter the position of insertion;\n";
cin>>n;
l1.insert(n);
} }while(ch=='y');
cout<<"The linked list after all insertions:\n";
l1.display();
do
{cout<<"Want to search a data:"<<endl;
cin>>ch;
if(ch=='y')
l1.search();
}while(ch=='y');
cout<<"The linked list after searching :";
```

```
l1.display();
return 1; }
```

Output:

```
Enter how many nodes u want to enter in linked list: 3
Enter first node data: 23
Enter node data: 7
Enter node data: 77
The data in linked list:
77 -> 7 -> 23 ->
Want to insert a node: y
Enter the position of insertion; 2
Enter data: 0
Want to insert a node: n
The linked list after all insertions:
The data in linked list:
77 -> 0 -> 7 -> 23 ->
Want to search a data: y
Enter data to be searched: 7
Data found:
Enter your choice:
1.delete data
2.replace data
 3.do nothing
2
Enter new data: 17
data replaced:
Want to search a data: y
Enter data to be searched:
```

23

Data found:

Enter your choice:

1.delete data

2.replace data

3.do nothing

1

Data deleted:

Want to search a data: n

The linked list after searching :The data in linked list:

77 -> 0 -> 17 ->

Result:

The above program has been executed successfully and output is verified.

Ex No:8 POLYNOMIAL ADDITION USING LINKED LIST

Date:

Aim:

To write a program to scan a polynomial using linked list and add two polynomial.

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: Create a structure poly with variables coeff , pow as int and *next as pointer

STEP 4: In the class add2poly create 3 variales of tpe poly nad initialise to NULL

STEP 5: In the function addploy() read the polynomials

STEP 6: Add the corresponding coefficients of same power

STEP 7: Display the added polynomial

STEP 8: Stop the process

Program Coding:

```
#include<iostream>
#include<iomanip>
using namespace std;
struct poly{
   int coeff;
   int pow;
   poly *next;
};
class add2poly
```

```
{ poly *poly1, *poly2, *poly3;
  public:
  add2poly(){poly1=poly2=poly3=NULL;}
  void addpoly()
{   int i,p;
     poly *newl=NULL,*end=NULL;
     cout<<"Enter highest power for x\n";
     cin>>p;
    cout<<"\nFirst Polynomial\n";
     for(i=p;i>=0;i--)
     {      newl=new poly;
     newl->pow=p;
     cout<<"Enter Co-efficient for degree"<<i<<":: ";
     cin>>newl->coeff;
     newl->next=NULL;
     if(poly1==NULL)
       poly1=newl;
     else
       end->next=newl;
     end=newl;      }
     cout<<"\n\nSecond Polynomial\n";
     end=NULL;
     for(i=p;i>=0;i--)
     {  newl=new poly;
     newl->pow=p;
     cout<<"Enter Co-efficient for+ degree"<<i<<":: ";
     cin>>newl->coeff;
     newl->next=NULL;
     if(poly2==NULL)
```

```
    poly2=newl;
  else
    end->next=newl;
  end=newl;       }
  poly *p1=poly1,*p2=poly2;
  end=NULL;
  while(p1 !=NULL && p2!=NULL){
   if(p1->pow == p2->pow){
      newl=new poly;
      newl->pow=p--;
      newl->coeff=p1->coeff + p2->coeff;
      newl->next=NULL;
      if(poly3==NULL)
        poly3=newl;
      else
        end->next=newl;
      end=newl;        }
    p1=p1->next;
    p2=p2->next;
  } }
void display()
{ poly *t=poly3;
cout<<"\n\nAnswer after addition is : ";
while(t!=NULL){
     if(t->coeff >0)
  cout<<"+"<<t->coeff;
  else
  cout<<t->coeff;
  cout<<"X^"<<t->pow;
```

```
      t=t->next;
    }
    cout<<endl;
    } };
int main(){
    add2poly obj;
     obj.addpoly();
     obj.display();
    return 1;
}
```

Output:

```
Enter highest power for x
3
First Polynomial
Enter Co-efficient for degree3::  -3
Enter Co-efficient for degree2::  0
Enter Co-efficient for degree1::  90
Enter Co-efficient for degree0::  3
Second Polynomial
Enter Co-efficient for degree3::  10
Enter Co-efficient for degree2::  11
Enter Co-efficient for degree1::  -85
Enter Co-efficient for degree0::  7
Answer after addition is : +7X^3+11X^2+5X^1+10X^0
```

Result:

The above program has been executed successfully and output is verified.

Ex No:9 LINEAR AND BINARY SEARCH

Date:

Aim:

To write a program to search an element from a list to perform Linear and Binary search. Use Template functions.

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: Create a template.

STEP 4: In the main method read an integer array of size 10.

STEP 5: the Lsearch() function does linear search

a. search the entire array in a for loop for the matching element
b. if found display the position(s)
c. else display element is not found

STEP 6: the Bsearch() function does the binary search.

a. The array is divided into two half
b. The middle value is compared with the element to searched
c. If its less than middle value the first half of the array is chosen to search
d. Else second part of the array is chosen for further search
e. Go to step a.
f. if found display the position(s)
g. else display element is not found

STEP 7: Stop the process

Program Coding:

```
#include<iostream>
using namespace std;
template <class T>
void Lsearch(T *a, T item, int n)
```

```
{ int z=0;
 for(int i=0;i<n;i++)
 {  if(a[i]== item)
  {   z=1;
   cout<<"\n Item found at position = "<<i+1<<"\n\n";  }
  else
   if(z!=1)
  {   z=0;  } }
 if(z==0)
  cout<<"\n Item not found in the list\n\n"; }
template <class T>
void Bsearch(T *a, T item, int n)
{ int beg=0,end=n-1;
 int mid=beg+end/2;
 while((a[mid]!=item) && (n>0))
 {  if(item>a[mid])
   beg=mid;
  else
   end=mid;
  mid=(beg+end)/2;
  n--;  }
 if(a[mid]==item)
  cout<<"\n Item found at position = "<<mid+1<<"\n\n";
 else
  cout<<"\n Item not found in the list\n\n";
}
 int main()
{ int iarr[10];
 int iele;
```

```
 cout<<"\n Enter  Elements of Integer Array \n";
for(int i=0;i<10;i++)
{  cin>>iarr[i]; }
cout<<"\n Elements of Integer Array \n";
for(int i=0;i<10;i++)
{  cout<<" "<<iarr[i]<<" ,"; }
cout<<"\n\n Enter an item to be search: ";
cin>>iele;
cout<<"\n\n Linear Search Method\n";
Lsearch(iarr,iele,10);
cout<<"\n\n Binary Search method\n";
Bsearch(iarr,iele,10);
return 1;
}
```

Output:

```
 Enter  Elements of Integer Array
23 42 43 48 50 56 61 78 83 95
 Elements of Integer Array
 23 , 42 , 43 , 48 , 50 , 56 , 61 , 78 , 83 , 95 ,
 Enter an item to be search: 78
 Linear Search Method
 Item found at position = 8

 Binary Search method
 Item found at position = 8
```

Result:

The above program has been executed successfully and output is verified.

Ex No:10 **SORTING**

Date:

Aim:

To write a program using templates to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: Create a template.

STEP 4: In the main method read an integer array of size 10.

STEP 5: Using Switch case statement select one appropriate operation

1. Bubble Sort
    a. Swap with value at location if its less than it
    b. Repeat until list is sorted
2. Selection sort
    a. Search the minimum element in the list
    b. Swap with value at location temp
    c. Increment temp to point to next element
    d. Repeat until list is sorted
3. Insertion sort
    a. If it is the first element, it is already sorted. return 1;
    b. Compare with all elements in the sorted sub-list
    c. Shift all the elements in the sorted sub-list that is greater than the value to be sorted
    d. Insert the value
    e. Repeat until list is sorted

STEP 6: Display the sorted list.

STEP 7: Stop the process

Program Coding:

```
#include<iostream>
using namespace std;
template <class T>
class sorting
{ T a[10];
public:
 void get_item()
 {  for(int i=0;i<10;i++)
  {   cout<<"\n a["<<i<<"] = ";
   cin>>a[i];
  } }
void sel_sort()
{ T temp;
 for(int i=0;i<10;i++)
 {  for(int j=i+1;j<10;j++)
  {   if(a[i]>a[j])
   {    temp=a[i];
    a[i]=a[j];
    a[j]=temp;
   } }
 }}
void bub_sort()
{ T temp;
 for(int i=0;i<10;i++)
 {  for(int j=0;j<10-i-1;j++)
  {   if(a[j]>a[j+1])
   {    temp=a[j];
    a[j]=a[j+1];
```

```
        a[j+1]=temp;
       }  } }
    }
    void inst_sort()
    { T tmp;
     int j;
     for(int i=1;i<10;i++)
     {  tmp=a[i];
      j=i-1;
      while(tmp<a[j])
      {   a[j+1]=a[j];
       j--;  }
      a[j+1]=tmp;
     }}
    void display()
    { for(int i=0;i<10;i++)
     {  cout<<" "<<a[i]<<", "; }
     cout<<"\n\n";
    }
    };// End of Class
    int main()
    {int ch;
     // Creating Integer Array using Template
     sorting<int> iarr;
     cout<<"\n\n Enter 10 Elements of Integer Array\n";
     iarr.get_item();
     cout<<"\n\n Elements of Integer Array\n";
     iarr.display();
     cout<<"\n1.BUBBLE SORT\n2.SELECTION SORT\n3.INSERTION
```

```
SORT\n4.EXIT\n";
 cout<<"Enter your Choice:"<<endl;
 cin>>ch;
 switch(ch)
 {case 1:iarr.bub_sort();
    cout<<"\n\n After Bubble Sorting\n    Elememts of Integer Array\n";
    iarr.display();
    break;
 case 2: iarr.sel_sort();
 cout<<"\n\n After Selection Sorting\n Elememts of Integer Array\n";
 iarr.display();
 case 3:iarr.inst_sort();
 cout<<"\n\n After Insertion Sorting\n Elememts of Integer Array\n";
 iarr.display();
case 4:break;
}
return 1;
 }
```

Output:

```
Enter 10 Elements of Integer Array
 a[0] = 5
 a[1] = 3
 a[2] = 8
 a[3] = 5
 a[4] = 9
 a[5] = 7
 a[6] = 2
 a[7] = 1
```

a[8] = 0

a[9] = 11

Elements of Integer Array

5, 3, 8, 5, 9, 7, 2, 1, 0, 11,

1.BUBBLE SORT

2.SELECTION SORT

3.INSERTION SORT

4.EXIT

Enter your Choice:

3

After Insertion Sorting

Elememts of Integer Array

1, 2, 3, 3, 5, 5, 7, 8, 9, 11,

Result:

The above program has been executed successfully and output is verified.

Ex No: 11 BINARY SEARCH TREE TRAVERSALS

Date:

Aim:

To write a program to create a Binary Search Tree and Display its preorder, postorder and inorder traversals Recursively

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: Create a class node with two pointers left and right and integer key.

STEP 4: Initialize key=-1 and left and right to NULL.

STEP 5: The setKey(),setLeft(),setRight() sets the respective values

STEP 6: The functions Key(),Left(),Right() gets the respective values

STEP 7: Class Tree is created with two node objects r and root.

STEP 8: The nodes are added to the tree with addnode() method.

STEP 9: The tree is traversed in inorder by recursion

a. inOrder(n->Left());
b. cout<<n->Key()<<" ";
c. inOrder(n->Right());

STEP 10: The tree is traversed in preorder by recursion

a. cout<<n->Key()<<" ";
b. preOrder(n->Left());
c. preOrder(n->Right());

STEP 11: The tree is traversed in postorder by recursion

a. postOrder(n->Left());
b. postOrder(n->Right());
c. cout<<n->Key()<<" ";

STEP 12: Delete the tree by using freenode() and delete() functions

STEP 13: Stop the process

Program Coding:

```
 #include<iostream>
using namespace std;
class node
{int key;
node* left;
node* right;
public:
node()
{key=-1;
left=NULL;
right=NULL;
};
void setKey(int aKey)
{key=aKey;};
void setLeft(node* aLeft)
{left=aLeft;};
void setRight(node* aRight)
{right=aRight;};
int Key()
{return key;};
node* Left()
{return left;};
node* Right()
{return right;};
};
class Tree
{node *r;
public:
```

```
node* Root()
{return r;};
void addnode(int key);
void inOrder(node* n);
void preOrder(node* n);
void postOrder(node* n);
private:
void addnode(int key,node* left);
void freenode(node* leaf);
};
Tree::Tree()
{r=NULL;}
Tree::~Tree()
{freenode(r);}
void Tree::freenode(node* leaf)
{if(leaf!=NULL)
{freenode(leaf->Left());
freenode(leaf->Right());
delete leaf;
}}
void Tree::addnode(int key)
{if(r==NULL)
{cout<<"\n Add root node..."<<key<<endl;
node* n=new node();
n->setKey(key);
r=n;
}
else
{cout<<"\n Add other node..."<<key<<endl;
```

```
addnode(key,r);
}}
void Tree::addnode(int key,node* leaf)
{if(key<=leaf->Key())
{if(leaf->Left()!=NULL)
addnode(key,leaf->Left());
else
{node* n=new node();
n->setKey(key);
leaf->setLeft(n);
}}
else
{if(leaf->Right()!=NULL)
addnode(key,leaf->Right());
else
{node* n=new node();
n->setKey(key);
leaf->setRight(n);
}}}
void Tree::inOrder(node* n)
{if(n)
{inOrder(n->Left());
cout<<n->Key()<<"   ";
inOrder(n->Right());
}}
void Tree::preOrder(node* n)
{if(n)
{cout<<n->Key()<<"   ";
preOrder(n->Left());
```

```
preOrder(n->Right());
}}
void Tree::postOrder(node* n)
{if(n)
{postOrder(n->Left());
postOrder(n->Right());
cout<<n->Key()<<"   ";
}}
int main()
{Tree* tree=new Tree();
tree->addnode(30);
tree->addnode(10);
tree->addnode(20);
tree->addnode(40);
tree->addnode(50);
cout<<"\n Inorder traversal: "<<endl;
tree->inOrder(tree->Root());
cout<<endl;
cout<<"\n Pre order traversal: "<<endl;
tree->preOrder(tree->Root());
cout<<endl;
cout<<"\n Post order traversal: "<<endl;
tree->postOrder(tree->Root());
cout<<endl;
delete tree;
return 1;
}
```

Output:

Add root node...30

Add other node...10

Add other node...20

Add other node...40

Add other node...50

Inorder traversal:

10 20 30 40 50

Pre order traversal:

30 10 20 40 50

Post order traversal:

20 10 50 40 30

Result:

The above program has been executed successfully and output is verified.

Ex No:12 DIAGONAL MATRIX

Date:

Aim:

To write a program to implement diagonal Matrix.

Algorithm:

STEP 1: Start the process

STEP 2: Include the necessary header files and namespace

STEP 3: In the main function declare variables r, c, nr, nc ,flag as int

STEP 4: Declare a two dimensional integer array x of size 10*10

STEP 5: Get the size of the r , c

STEP 6: If r!=c then display not a square matrix and stop the process

STEP 7: Else, get input for the matrix inside a nested for loop

STEP 8: Display the matrix in matrix format.

STEP 9: Set flag=1

STEP 10: Inside a nested for loop

a. Check if the diagonal elements where r=c are zero
b. If so, set flag=0
c. Check if non diagonal elements are non zero
d. If so set flag=0

STEP 11: Display the given matrix is diagonal if flag is 1

STEP 12: Else display the given matrix is not diagonal

STEP 13: Stop the process

Program coding:

```
#include<iostream>
using namespace std;
int main()
```

```
{
  int  x[10][10] , nr , nc , r , c , flag ;
cout<<"Enter the number of rows and columns: ";
cin>>nr;
cin>>nc;
  if(nr==nc)                    /* checking for square matrix */
  {    cout<<"Enter elements of the matrix: \n";
    for(r=0 ; r<nr ; r++)
      for(c=0 ; c<nc ; c++)
        cin>>x[r][c];
    flag=1 ;
    for(r=0 ; r<nr ; r++)
      for(c=0 ; c<nc ; c++)
        if(r==c) /* true for diagonal elements */
        {    if(x[r][c]==0)
                 flag=0;
        }
        else
        {  if(x[r][c]!=0)
                flag=0;
        }
for(r=0 ; r<nr ; r++)
{for(c=0 ; c<nc ; c++)
{cout<<x[r][c]<<"  ";}
cout<<endl;}
    if(flag==1)
      cout<<"The matrix is diagonal";
    else
      cout<<"The matrix is not diagonal";
```

```
  }
  else
    cout<<"The matrix is not a sqaure matrix";
return 0;
}
```

Output:

```
Enter the number of rows and columns: 3 4
The matrix is not a sqaure matrix
Enter the number of rows and columns: 3 3
Enter elements of the matrix:
1 2 3
2 1 3
0 0 4
1  2  3
2  1  3
0  0  4
The matrix is not diagonal
Enter the number of rows and columns: 3 3
Enter elements of the matrix:
1 0 0
0 2 0
0 0 3
1  0  0
0  2  0
0  0  3
The matrix is diagonal
```

Result: The above program has been executed successfully and output is verified.