Lab Questions

Lab 1

1. **WAP to check whether the entered number is prime or not using POP and OOP way.**
2. **WAP to calculate the net salary of an employee under the following condition: a) If salary > 25000, Bonus is 15% b) If salary > 20,000, Bous is 10% c) Else bonus is 5%**
3. **Write a temperature conversion program from celcius to farhenheit and vice-versa depending on choice by the user.**

Lab2

1. **WAP to find the area of the triangle(when three sides are given) and the area of the triangle(when base and height are given) using function overloading and default argument.**
2. **WAP to find the volume of the cube, cuboid and cylinder using function overloading and default arguments.**
3. **Write meaningful program to show use of default arguments.**
4. **WAP to calculate interest amount when principle, time and rate are given. Use concept of inline function and default argument.**
5. **WAP that shows the concept of pass by reference.**
6. **WAP with function that takes two arguments as reference and return the smaller one by reference. In main program, take two numbers as input from users and find the average of the two numbers. Call the function by assigning the average value to the function and display the value of both arguments.**

7. **WAP using new and delete operators to store n numbers dynamically and find the average of the numbers by using casting operator.**

Lab3

1. **WAP that shows the concept of namespace.**
2. **WAP with function that uses pass by reference to change meter to centimeter. Use pass by reference along with the namespace.**
3. **WAP to show the use of Enumeration data type.**
4. **WAP with class to represent time having member functions to read and display time. Your program should be designed in such a way that the member function to read time should be declared within the class definition and should be defined outside of the class definition. Member function to display time should be declared within the class definition.**
5. **Write a temperature conversion program to convert celcius to Fahrenheit and vice versa. The choice entered by the user. Use concept of class and object.**

Lab4

1. **WAP with a class to represent rectangle. The program should have constructors and other member functions to customise areas and perimeter and display area and perimeter.**
2. **Create a class complex with two member variables real and imaginary of type float. Write default, parameterized and copy constructors. Make necessary function to display state of the object on the screen.**
3. **WAP that can store DepartmentID and DepartmentName with constructor. Also write destructor in the same**

class and show that the objects are destroyed in the reverse order of creation with suitable message.

4. **Write a complete program that illustrates the object concept to add and subtract time units . A time unit has hour, minute and second as member. The memeber functions to add and subtract time should have objects as an argument.**
5. **WAP with class to represent complex number. The program should be able to find sum of the two complex numbers. The member function to calculate sum should have two objects as an argument.**

Lab5

1. **WAP that will have a class with members name, roll number, directory, filename and other member functions as required. All of the objects will share two members directory and filename where the information of student is stored.**
2. **WAP that will allocate memory dynamically for the pointer member of the string class that point to character array. Write a meaningful program for string manipulation.**
3. **Write a class with member variable, roll, name and marks in 5 subjects as an array of float and a member function to display the state of the object on the source. Use DMA to allocate memory for second member variable in a default and parameterized constructor. Also show how a pointer to object can be used to call the member function.**
4. **Write a meaningful program that shows the use of static data member and static member function.**
5. **WAP that shows the use of constant member function and constant object.**

6. **WAP for DMA for array of object and one single object.**
7. **WAP to add and subtract two distances by using the concept of object as an argument and returning the object by the function. The distance has feet and inches as the member.**

Lab 6

1. **WAP using friend function to compare the data member of two diffferent classes and display the largest one.**
2. **WAP to add member of objects of two different classes using friend function.**
3. **Create a class mdistance to store the values in meter and centimeter and class edistance to store the values in feet and inches. Perform addition of object of mdistance and addition of object od edistance using friend function.**
4. **WAP that show how the member function of one class can be friend function of another class.**
5. **Write a meaningful program that shows the concept of friend class.**

Lab 7

1. **WAP to overload unary prefix(++) using member operator function and unary prefix(--) using friend function.**
2. **WAP to concatenate two strings using the concept of operator overloading (+).**
3. **WAP to compare the magnitude of complex numbers by overloading <, > and == operator.**
4. **WAP that will add objects of a date class with members day, month and year using operator**

overloading. Make another function to find no. of days in between two dates by operator overloading.

5. **WAP to define a class Distance with necessay data members and functions. Then overload relational operators to compare two objects of distance class.**
6. **WAP having a class to represent money. The class should have two integer members to represent rupees and paisa. Overload + and - for adding and subtracting the objects. Then overload >, < , == and != operator for comparing objects.**
7. **Write a class having an array as member. Overload the index operator([]) to input and display the elements.**

Lab 8

1. **WAP to add two matrices by overloading + operator.**
2. **WAP to achieve operation like c2 = 2 + c1, where c1 and c2 are complex numbers. Also for c2 = c1 + 2 using operator overloading.**
3. **WAP that shows the conversion from basic type to class type.**
4. **WAP that shows the conversion from class type to basic type.**
5. **WAP to convert object of a class that represents weight of gold in tola to object of class that represents weight in grams. (1 tola = 1.664 gm)**
6. **WAP to convert type of object of polar class to rectangle class and vice versa. Apply constructor functions in both the class.**

Lab 9

1. **WAP that will inherit a class where member are increased by its member function. The derived class should override the function that increase its**

member in the derived class that will add increment of the derived class data member. Create other meaningful function that suit your requirement.

2. **WAP to show ambiguity and its resolution in multiple inheritance.**
3. **WAP with three classes student, test and result by using multilevel inheritance. Assume necessary data members and functions yourself and program with input information, input data and calculate marks, total and display the result.**
4. **WAP with a class cricketer that has data member to repesent name, age and number of match played. From this class cricketer, derive two classes, bowler and batsman. The bowler class should have number of wicket as data member and batsman class should have number of runs and number of centuries as data members. Use appropriate member functions in all classes to make your program meaningful.**
5. **WAP that shows the concept of hybrid inheritance.**
6. **Define a class publication which has a title. Derive two classes from it. A class book which has an accession number and a class magazine which has volume number. With these two as bases, derive the class journal. Define a function print() in each of these classes. Ensure that the derived class function always invokes the base class function. In main, create a journal called IEEOOP with an accession number 681.3 and a volume number 1. Invoke the print() function for this object.**

Lab 10

1. **WAP that shows runtime polymorphism in C++.**
2. **WAP having student as an abstract class and create derived class such as engineering, science and**

medical. Create their objects, process them and access them using base class pointer.

3. **WAP that shows the use of pure virtual function.**
4. **A base class pointer contains address of its derived class object and if base class pointer goes out of scope, which destructor is called. WAP to illustrate this.**
5. **WAP that shows RTTI mechanism in C++. a) dynamic_cast b) typeid**

Lab 11

1. **WAP with class template to represent array and add member functions to find minimum, maximum and sort the generic array.**
2. **WAP for finding the sum and average of an array elements using function template.**
3. **Write a function template for the function power() which has two parameters base and exp and returns base^exp. the type of base is the parameter to the template and type of exp is int. If the exp is negative then it must be converted to its positive equivalent.**
4. **Create a class template with necessary member variables to represent vectors. Write initialization mechanism and overload * operator to find the scalar product of the two vectors. Also write a member function to display the vector. Use above template and write main funciton to perform scalar product of vectors of type int and double.**
5. **Define a class to represent time. It should have memvber function to read and display the time. The function to read time should raise an exception if the user enter invalid values for hour, minutes and seconds. The exception thrown should contain an**

argument. The exception thrown should be thandled outside of the member functions of the class.

6. Write a meaningful program that can handle multiple exceptions.
7. Write a meaningful program to use both exception specification for function.

Lab 1

1. WAP to check whether the entered number is prime or not using POP and OOP way.

```
#include <iostream>
using namespace std;
int main()
{
    int count = 0;
    cout << "Enter the number: ";
    int n;
    cin >> n;
    for(int i=1;i<=n;i++)
    {
        if(n%i==0)
            count++;
    }
   if(count==2 || count == 1)
        cout << n << " is prime.";
    else
        cout << n << " is not prime.";
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int count = 0;
    cout << "Enter the number: ";
    int n;
    cin >> n;
```

```
    for(int i=1;i<=n;i++)
   {
        if(n%i==0)
            count++;
    }
    if(count==2 || count == 1)
        cout << n << " is prime.";
    else
        cout << n << " is not prime.";
    return 0;
}
```

2. **WAP to calculate the net salary of an employee under the following condition: a) If salary > 25000, Bonus is 15% b) If salary > 20,000, Bous is 10% c) Else bonus is 5%**

```
#include <iostream>
using namespace std;
int main()
{
    float salary;
    cout << "Enter the salary: ";
    cin >> salary;
    float bonus = 0.0;
    if(salary > 25000)
        bonus  = 0.15 * salary;
    else if(salary>20000 && salary<=25000)
        bonus = 0.10*salary;
    else
        bonus = 0.05*salary;
    cout << "The salary with bonus is: " <<
(bonus+salary) << endl;
    return 0;
}
```

3. **Write a temperature conversion program from celcius to farhenheit and vice-versa depending on choice by the user.**

```
#include <iostream>
```

```
using namespace std;
int main()
{
    int n,c,f;
    cout << "Enter your choice\n";
    cout << "1. Celcius to Fahrenheit" << endl
         << "2. Farhenheit to Celcius" << endl;
    cin >> n;
    switch(n) {
        case 1: cout << "Enter the temp in celcius: ";
                cin >> c;
                f  = 1.8 * c + 32;
                cout << "Fahrenheit = " << f;
                break;
        case 2: cout << "Enter the temp in
fahrenheit:";
                cin >> f;
                c = (f-32)/1.8;
                cout << "Celcius = " << c;
                break;
        default: cout << "Invalid choice";
    }
    return 0;
}
```

Lab2

1. **WAP to find the area of the triangle(when three sides are given) and the area of the triangle(when base and height are given) using function overloading and default argument.**

```
#include <iostream>
#include <cmath>
using namespace std;
void area(float a, float b, float c)
   {
    if((a+b)>c && (b+c)>a && (c+a)>b)
   {
        float s = (a + b + c) / 2;
        float area = sqrt(s * (s-a) * (s-b) * (s-c));
```

```
        cout << "Area is: " << area << endl;
    } else {
        cout << "The area can't be computed" << endl;
           }
}
void area(float base, float height = 1)
{
    float area;
    area = 0.5 * base * height;
    cout << "Area is: " << area << endl;
}
int main()
 {
    float a,b,c;
    cout << "Enter the sides a,b,c" << endl;
    cin >> a >> b >> c;
    area(a,b,c);
    cout << "Enter the base and height" << endl;
    float base,height;
    cin >> base >> height;
    area(base,height);
    area(base);
    return 0;
}
```

2. **WAP to find the volume of the cube, cuboid and cylinder using function overloading and default arguments.**

```
#include <iostream>
#define PI 3.14
using namespace std;
void volume(int l)
 {
    cout << "Volume of cube is " << l*l*l << endl;
}
void volume(int l, int b, int h) {
    cout << "Volume of cuboid is " << l*b*h << endl;
}
void volume(float r, int h) {
    cout << "Volume of cylinder is " << PI*r*r*h <<
endl;
```

```
}
int main()
{
    volume(5);
    volume(5,2,6);
    volume(3,5);
    return 0;
}
```

3. **Write meaningful program to show use of default arguments.**

```
#include <iostream>
using namespace std;
int area(int l)
 {
    return l*l;
 }
int area(int b, int h)
{
    return b*h;
}
float area(float r)
{
    return 3.14*r*r;
}
int main() {
    cout << area(9) << endl;
    cout << area(5) << endl;
    cout << area(5,6) << endl;
    return 0;
}
```

4. **WAP to calculate interest amount when principle, time and rate are given. Use concept of inline function and default argument.**

```
#include <iostream>
using namespace std;
inline void interest(int p, int t, float r = 0.1)
 {
    cout << "Interest = " << p*t*r << endl;
 }
```

```
int main()
{
   interest(1000,2);
    interest(1000,2,0.2);
    return 0;
}
```

5. **WAP that shows the concept of pass by reference.**

```
#include <iostream>
using namespace std;
inline void swap(int &a, int &b)
{
    int temp = b;
    b = a;
    a = temp;
}
int main()
 {
    int n1 = 1, n2 = 2;
    cout << "Before swapping, the values are:" << endl;
    cout << n1 << "\t" << n2 << endl;
    swap(n1,n2);
    cout << "After swapping, the values are:" << endl;
    cout << n1 << "\t" << n2 << endl;
    return 0;
}
```

6. **WAP with function that takes two arguments as reference and return the smaller one by reference. In main program, take two numbers as input from users and find the average of the two numbers. Call the function by assigning the average value to the function and display the value of both arguments.**

```
#include <iostream>
using namespace std;
float &small(float &a, float &b)
 {
    return a < b ? a : b;
}
int main()
{
```

```
    float n1 = 2.0, n2 = 4.0;
    float average = (n1+n2)/2;
    cout << "The values before are: " << n1 << " " <<
n2 << endl;
    small(n1,n2) = average;
    cout << "The values after are: " << n1 << " " << n2
<< endl;
    return 0;
}
```

7. **WAP using new and delete operators to store n numbers dynamically and find the average of the numbers by using casting operator.**

```
#include <iostream>
using namespace std;
int main()
 {
    int n, *ptr, sum = 0;
    float avg;
    cout << "How manu numbers:";
    cin >> n;
    ptr = new int[n];
    for(int i=0;i<n;i++)
    {
        cin >> ptr[i];
        sum += ptr[i];
    }
    avg = static_cast<float>(sum)/n;
    cout << "\nThe average is: " << avg;
    delete []ptr;
    return 0;
}
```

Lab3

1. **WAP that shows the concept of namespace.**

```
#include <iostream>
using namespace std;
namespace declare
{
    char name[25];
```

```
    int roll;
}
namespace result
{
    int marks;
}
void output()
{
    using declare::name;
    cout << "Name: " << name << endl
         << "Roll no: " << declare::roll << endl
         << "Marks: " << result::marks << endl;
}
int main()
{
    using namespace declare;
    cout << "Enter the name:";
    cin >> name;
    cout << "Enter the roll no.";
    cin >> roll;
    cout << "Enter the marks:";
    cin >> result::marks;
    output();
}
```

2. **WAP with function that uses pass by reference to change meter to centimeter. Use pass by reference along with the namespace.**

```
#include <iostream>
using namespace std;
namespace length
{
    int m, cm;
}
void convert(int &x, int &y)
{
    y = 100 * x;
}
int main()
 {
    using namespace length;
```

```
    cout << "Enter the length in meter:\n";
    cin >> m;
    convert(m, cm);
    cout << "The length in centimeter is: " << cm;
    return 0;
}
```

3. **WAP to show the use of Enumeration data type.**

```
#include <iostream>
using namespace std;
enum days {sun=1,mon, tue, wed, thu, fri, sat};
int main()
 {
    int n;
    cout << "Enter the day of week: ";
    cin >> n;
    switch(n) {
          case sun: cout << "Sunday";
                  break;
          case mon: cout << "Monday";
                    break;
          case tue: cout << "Tuesday";
                    break;
          case wed: cout << "Wednesday";
                    break;
          case thu: cout << "Thursday";
                    break;
          case fri: cout << "Friday";
                    break;
          case sat: cout << "Saturday";
                    break;
          default: cout << "Invalid choice";
    }
    return 0;
}
```

4. **WAP with class to represent time having member functions to read and display time. Your program should be designed in such a way that the member function to read time should be declared within the class definition and should be defined outside of**

the class definition. Member function to display time should be declared within the class definition.

```
#include <iostream>
using namespace std;
class Time
{
    int h,m,s;
    public:
    void read();
    void display()
{
            cout << "Time is: " << h << ": " << m <<
" : " << s << endl;
}
};
void Time::read()
 {
    cout << "Enter the Time in hh:mm::sec format.";
    cin >> h >> m >> s;
 }
int main()
{
    Time t;
    t.read();
    t.display();
    return 0;
}
```

5. Write a temperature conversion program to convert celcius to Fahrenheit and vice versa. The choice entered by the user. Use concept of class and object.

```
#include <iostream>
using namespace std;
class temp
{
    float c,f;
    public:
        void input(int n)
```

```
{
            if(n==1)
 {
                cout << "Enter the celcius:";
                cin >>c;
 }
            else
{
                cout << "Enter the fahrenheit:";
                cin >> f;
}
}
void convert(int n)
{
            if(n==1)
 {
                f = 1.8*c+32;
                cout << "Fahrenheit is: " << f << endl;
 } else
 {

                c = (f-32)/1.8;
               cout << "Celcius is: " << c;
 }
}
};
int main()
 {
    cout << "Choose:" << endl
         << "1. Celcius to Farhenheit" << endl
         << "2. Fahrenheit to Celcius" << endl;
    int n;
    cin >> n;
    temp t;
    switch(n)
{
        case 1:
        case 2: t.input(n);
                t.convert(n);
                break;
```

```
        default: cout << "Invalid choice" << endl;
    }
    return 0;
}
```

Lab4

1. **WAP with a class to represent rectangle. The program should have constructors and other member functions to customise areas and perimeter and display area and perimeter.**

```
#include <iostream>
using namespace std;
class Rectangle {
    float l,b;
    public:
        Rectangle() {
            cout << "Enter the length:";
            cin >> l;
            cout << "Enter the breadth:";
            cin >> b;
            }
        Rectangle(int x, int y) {
            l = x;
            b = y;
        }
        void area() {
            cout << "Area is: " << l*b << endl;
        }
        void perimeter() {
             cout << "Perimeter is: " << 2*(l+b);
        }
};
int main() {
    Rectangle r;
    r.area();
    r.perimeter();
    cout << "\nParemeterized constructor:" << endl;
    Rectangle rec(2,4);
    rec.area();
    rec.perimeter();
```

```
    return 0;
}
```

2. **Create a class complex with two member variables real and imaginary of type float. Write default, parameterized and copy constructors. Make necessary function to display state of the object on the screen.**

```
#include <iostream>
using namespace std;
class Department {
    int DepartmentID;
    char DepartmentName[25];
    public:
        Department() {
            cout << "Enter the department id: ";
            cin >> DepartmentID;
            cout << "\nEnter the department name: ";
            cin >> DepartmentName;
        }
        ~Department(){cout << "Destroyed department
with id:" << DepartmentID << endl;}
};
int main() {
   Department d1,d2,d3;
    return 0;
}
```

3. **WAP that can store DepartmentID and DepartmentName with constructor. Also write destructor in the same class and show that the objects are destroyed in the reverse order of creation with suitable message.**

```
#include <iostream>
using namespace std;
class Department {
    int DepartmentID;
    char DepartmentName[25];
    public:
        Department() {
            cout << "Enter the department id: ";
```

```
            cin >> DepartmentID;
            cout << "\nEnter the department name: ";
            cin >> DepartmentName;
        }
        ~Department(){cout << "Destroyed department
with id:" << DepartmentID << endl;}
};
int main() {
    Department d1,d2,d3;
    return 0;
}
```

4. **Write a complete program that illustrates the object concept to add and subtract time units . A time unit has hour, minute and second as member. The memeber functions to add and subtract time should have objects as an argument.**

```
#include <iostream>
using namespace std;
class times {
    int hr,m,sec;
    public:
        void add(times &x, times &y) {
            int total1,total2,sum;
            total1 = x.hr *3600 + x.m *60 + x.sec;
            total2 = y.hr *3600 + y.m *60 + y.sec;
            sum = total1+total2;
            hr = sum/3600;
            m = (sum%3600) / 60;
            sec = (sum%3600) % 60;
        }
        void subtract(times &x, times &y) {
            int total1,total2,diff;
            total1 = x.hr *3600 + x.m *60 + x.sec;
            total2 = y.hr *3600 + y.m *60 + y.sec;
            if (total1>total2)
                diff = total1-total2;
            else
            diff = total2-total1;
            hr = diff/3600;
```

```
            m = (diff%3600) / 60;
            sec = (diff%3600) % 60;
        }
       void input() {
            cout << "Enter the time in hr:m:sec ";
            cin >> hr >> m >> sec;
        }
        void display() {
            cout << "The time is " << hr << ":" << m <<
":" << sec;
        }
     ~times() {}
};
int main() {
    times t1,t2,t3;
    t1.input();
    t2.input();
    cout << "On adding:\n";
    t3.add(t1,t2);
    t3.display();
    cout << "\nOn subtracting" << endl;
    t3.subtract(t1,t2);
    t3.display();
    return 0;
}
```

5. **WAP with class to represent complex number. The program should be able to find sum of the two complex numbers. The member function to calculate sum should have two objects as an argument.**

```
#include <iostream>
using namespace std;
class complex {
    float real,imag;
    public:
        void input() {
            cout << "Enter the real part: ";
            cin >> real;
            cout << "Enter the imaginary part: ";
```

```
            cin >> imag;
        }
        void display() {
            cout << "\nThe complex number is " << real
<< "+" << imag << "j";
        }
        void add(complex x, complex y) {
            cout << "Adding the complex numbers";
            real = x.real + y.real;
            imag = x.imag + y.imag;
        }
};
int main() {
    complex c1,c2,c3;
    c1.input();
    c2.input();
    c3.add(c1,c2);
    c3.display();
    return 0;
}
```

Lab5

1. **WAP that will have a class with members name, roll number, directory, filename and other member functions as required. All of the objects will share two members directory and filename where the information of student is stored.**

```
#include <iostream>
using namespace std;
class student {
    char name[30];
    int rollno;
    public:
        static char directory[100];
        static char filename[50];
        void input() {
            cout << "Enter the name:";
            cin >> name;
            cout << "Enter the roll no:";
```

```
            cin >> rollno;
        }
        void output() {
            cout << "\n Name:" << name
                 << "\n Roll no:" << rollno
                 << "\n Directory:" << directory
                 << "\n Filename:" << filename;
        }

};
char student::filename[50] = "2071.txt";
char student::directory[100] = "/KEC/";
int main() {
    student s1;
    s1.input();
    s1.output();
    return 0;
}
```

2. **WAP that will allocate memory dynamically for the pointer member of the string class that point to character array. Write a meaningful program for string manipulation.**

```
#include <iostream>
#include <string.h>
using namespace std;
class strings {
    char *name;
    public:
        strings() {}
        strings (char *DN) {
            int l = strlen(DN);
            name = new char[l];
            strcpy(name,DN);
        }
        strings concat (strings st) {
            strings temp;
            int length = strlen(st.name)+ strlen(name);
            temp.name = new char[length];
            strcpy(temp.name,name);
            strcat(temp.name,st.name);
```

```
            return temp;
        }
        void display() {
            cout << "The concatenated string is: " <<
name;
        }
};
int main() {
    strings s1("Engineer"),s2("ing"),s;
    s = s1.concat(s2);
    s.display();
    return 0;
    return 0;
}
```

3. **Write a class with member variable, roll, name and marks in 5 subjects as an array of float and a member function to display the state of the object on the source. Use DMA to allocate memory for second member variable in a default and parameterized constructor. Also show how a pointer to object can be used to call the member function.**

```
#include <iostream>
#include <string.h>
using namespace std;
class student {
    int roll;
    char *name;
    float marks[5];
    public:
        student() {
            name = new char[1];
        }
       student(char *n) {
            int l = strlen(n);
            name = new char[l];
            strcpy(name,n);
        }
        void read() {
            cout << "Enter the name:";
```

```
            cin >> name;
            cout << "Enter the roll no.";
            cin >> roll;
             cout << "Enter the marks of the subjects"
<< endl;
            for (int i=0;i<5;i++) {
                cout << "Marks "<< i+1 <<":";
                cin >> marks[i];
            }
        }
        void display() {
            cout << "\nName:" << name;
            cout << "\nRoll no:" << roll << endl;
            for (int j=0;j<5;j++)
                cout << "Marks of subject["<< j+1<<"] =
" << marks[j] << endl;
        }
};
int main() {
    student *s;
    s = new student("amit");
    s->read();
    s->display();
    return 0;
}
```

4. **Write a meaningful program that shows the use of static data member and static member function.**

```
#include <iostream>
using namespace std;
class demo {
    static int count;
    public:
        demo() {
            cout << "Created a object" <<endl;
            count++;
        }
        static void displaycount()
        {
            cout << "Count=" << count << endl;
        }
```

```
};
int demo::count = 0;
int main()
{
    demo d1;
    demo::displaycount();
    demo d2;
    demo::displaycount();
    demo d3;
    demo::displaycount();
    return 0;
}
```

5. **WAP that shows the use of constant member function and constant object.**

```
#include <iostream>
using namespace std;
class number {
    int n;
    const int count = 0;
    public:
        number(int x) {
            cout << "Created a object" << endl;
            n = x;
        }
        void increasenumber() {
            n++;
        }
        void displaycount() const
        {
            cout << "Constant member function." <<
endl;
            cout << "Count=" << count << endl;
        }
        void displaynumber() const {
            cout << "Number is:" << n << endl;
        }
        void displayboth()
        {
            cout << "Count=" << count
                 << " Number = " << n << endl;
```

```
        }
};
int main()
{
    number d1(1);
    cout << "Normal object:" << endl;
    d1.displaycount();
    d1.displayboth();
    d1.increasenumber();
    d1.displayboth();
    cout << "\nConstant object" << endl;
    const number d2(5);
    d2.displaycount();
    d2.displaynumber();
    // d2.displayboth(); will give error
    return 0;
}
```

6. **WAP for DMA for array of object and one single object.**

```
#include <iostream>
using namespace std;
const int SIZE = 3;
class marks {
    float mark;
    public:
        void input() {
            cout << "Enter the marks" << endl;
            cin >> mark;
        }
        void output() {
            cout <<  "Marks = " << mark << endl;
        }
};
int main() {
    marks *m, *array_marks;
    m = new marks;
    array_marks = new marks[SIZE];
    cout << "DMA for single object" << endl;
```

```
    m->input();
    m->output();
    cout << "DMA for the array of object" << endl;
    for (int i=0;i<SIZE;i++) {
        array_marks[i].input();
        array_marks[i].output();
    }
    return 0;
}
```

7. **WAP to add and subtract two distances by using the concept of object as an argument and returning the object by the function. The distance has feet and inches as the member.**

```
#include <iostream>
using namespace std;
class distances {
    int feet,inches;
    public:
    void input() {
    cout << "Enter the feet and inches";
    cin >> feet >> inches;
    }
    void display() {
        cout << feet << "feet " << inches << "
inches"<< endl;
   }
    distances adddistance(distances d1,distances d2) {
        distances temp;
        temp.inches = d1.inches + d2.inches;
        temp.feet  = d1.feet + d2.feet +
temp.inches/12;
        temp.inches = temp.inches%12;
        return temp;
    }
    distances subtractdistance (distances d1, distances
d2) {
        distances temp;
        temp.feet  = 0;
```

```
        temp.inches = d2.inches - d1.inches;
        if (temp.inches<0)
        {
            temp.inches = temp.inches +12;
            temp.feet = temp.feet-1;
        }
        temp.feet = temp.feet + d2.feet-d1.feet;
        return temp;
    }
};
int main() {
    distances d1,d2,d3,d4;
    d1.input();
    d2.input();
    cout << "On adding: ";
    d3.adddistance(d1,d2).display();
    cout << "On subtracting: ";
    d4.subtractdistance(d1,d2).display();
    return 0;
}
```

Lab 6

1. **WAP using friend function to compare the data member of two diffferent classes and display the largest one.**

```
#include <iostream>
using namespace std;
class second;
class first {
    int data1;
    public:
        first(int x = 0): data1(x) {}
        friend void large(first, second);
        ~first() {}
};
class second {
    int data2;
    public:
        second(int y=0): data2(y){}
```

```
        friend void large(first, second);
        ~second() {}
};
void large(first a, second b) {
    cout << "The largest is: " << (a.data1>b.data2 ?
a.data1:b.data2);
}
int main() {
    first f(1);
    second s(2);
    large(f,s);
    return 0;
}
```

2. **WAP to add member of objects of two different classes using friend function.**

```
#include <iostream>
#include <string.h>
using namespace std;
class String {
    char *str;
    public:
        String(char *s) {
            int length = strlen(s);
            str = new char[length];
            strcpy(str, s);
        }
       String() {}
        String operator+(String s) {
            String temp;
            int l = strlen(str) + strlen(s.str);
            temp.str = new char[l];
            strcpy(temp.str, str);
            strcat(temp.str, s.str);
            return temp;
        }
        void display() {
            cout << str << endl;
        }
};
int main() {
```

```
    String s1("Amit"), s2("Chaudhary"), s3;
    s1.display();
    s2.display();
    cout << "After concatenation: ";
    s3 = s1 + s2;
    s3.display();
    return 0;
}
```

3. Create a class mdistance to store the values in meter and centimeter and class edistance to store the values in feet and inches. Perform addition of object of mdistance and addition of object od edistance using friend function.

```
#include <iostream>
using namespace std;
class second;
class first {
    int data1;
    public:
        first(int x = 0): data1(x) {}
        friend void large(first, second);
        ~first() {}
};
class second {
    int data2;
    public:
        second(int y=0): data2(y){}
        friend void large(first, second);
        ~second() {}
};
void large(first a, second b) {
    cout << "The largest is: " << (a.data1>b.data2 ?
a.data1:b.data2);
}
int main() {
    first f(1);
    second s(2);
    large(f,s);
    return 0;
}
```

4. **WAP that show how the member function of one class can be friend function of another class.**

```
#include <iostream>
using namespace std;
class second;
class first {
    int data1;
    public:
        first(int x = 0): data1(x) {}
        friend void large(first, second);
        ~first() {}
};
class second {
    int data2;
    public:
        second(int y=0): data2(y){}
        friend void large(first, second);
        ~second() {}
};
void large(first a, second b) {
    cout << "The largest is: " << (a.data1>b.data2 ?
a.data1:b.data2);
}
int main() {
    first f(1);
    second s(2);
    large(f,s);
    return 0;
}
```

5. **Write a meaningful program that shows the concept of friend class.**

```
#include <iostream>
using namespace std;
class second;
class first {
    int data1;
    public:
```

```
        first(int x = 0): data1(x) {}
        friend void large(first, second);
        ~first() {}
};
class second {
    int data2;
    public:
        second(int y=0): data2(y){}
        friend void large(first, second);
        ~second() {}
};
void large(first a, second b) {
cout << "The largest is: " << (a.data1>b.data2 ?
a.data1:b.data2);
}
int main() {
    first f(1);
    second s(2);
    large(f,s);
    return 0;
}
```

Lab 7

1. **WAP to overload unary prefix(++) using member operator function and unary prefix(--) using friend function.**

```
#include <iostream>
using namespace std;
class number {
    int n;
    public:
        void operator++() {
            ++n;
        }
        void display() {
            cout << n << endl;
        }
        void getinput() {
            cout << "Enter the number: ";
```

```
            cin >> n;
        }
        friend void operator --(number &);
};
void operator--(number &num) {
    --num.n;
}
int main() {
    number no;
    no.getinput();
    cout << "Before ++ operation" << endl;
    no.display();
    ++no;
    cout << "After ++ operation" << endl;
    no.display();
    cout << "After -- operation" << endl;
    --no;
    no.display();
    return 0;
}
```

2. **WAP to concatenate two strings using the concept of operator overloading (+).**

```
#include <iostream>
#include <string.h>
using namespace std;
class String {
    char *str;
    public:
        String(char *s) {
            int length = strlen(s);
            str = new char[length];
            strcpy(str, s);
        }
        String() {}
        String operator+(String s) {
            String temp;
            int l = strlen(str) + strlen(s.str);
            temp.str = new char[l];
            strcpy(temp.str, str);
            strcat(temp.str, s.str);
```

```
            return temp;
        }
        void display() {
            cout << str << endl;
        }
};
int main() {
    String s1("Amit"), s2("Chaudhary"), s3;
    s1.display();
    s2.display();
    cout << "After concatenation: ";
    s3 = s1 + s2;
    s3.display();
    return 0;
}
```

3. **WAP to compare the magnitude of complex numbers by overloading <, > and == operator.**

```
#include <iostream>
#include <cmath>
using namespace std;
class complex {
    float real,imag;
    public:
        complex(float r = 0, float i=0): real(r),
imag(i) {}
        float magnitude() {
            float m = sqrt(real*real + imag*imag);
            return m;
        }
        void input() {
            cout << "Enter the real and imaginary part"
<< endl;
            cin >> real >> imag;
        }
        void display() {
            cout << "(" << real << "," << imag << ")";
        }
        int operator==(complex c2) {
            if(magnitude()==c2.magnitude())
                return 1;
```

```
            return 0;
        }
        int operator<(complex c2) {
            if(magnitude()<c2.magnitude())
                return 1;
            return 0;
        }
        int operator>(complex c2) {
            if(magnitude()>c2.magnitude())
                return 1;
            return 0;
        }
};
int main() {
complex c1,c2;
c1.input();
c2.input();
if(c1==c2) {
cout << "They are equal" << endl;
} else if (c1>c2) {
        c1.display();
        cout << " is greater than";
        c2.display();
        cout << endl;
    } else if(c1<c2) {
        c1.display();
        cout << " is less than";
        c2.display();
        cout << endl;
    }
    return 0;
}
```

4. **WAP that will add objects of a date class with members day, month and year using operator overloading. Make another function to find no. of days in between two dates by operator overloading.**

```
#include <iostream>
using namespace std;
class Date {
    int day,m,y;
```

```
    public:
        Date(int p=0, int q=0, int r=0) {
            day = p;
            m = q;
            y = r;
        }
        void display() {
            cout << "(" << day << "/" << m << "/" << y
<< ")";
        }
        Date operator+(Date d) {
            Date temp;
            temp.day  = day + d.day;
            temp.m = m + d.m + temp.day/30;
            temp.day %= 30;
            temp.y = y+ d.y + temp.m/12;
            temp.m %= 12;
            return temp;
        }
        int operator-(Date d) {
            int days;
            int days1 = day+m*30+y*365;
            int days2 = day+m*30+y*365;
            days = days2 - days1;
            if(days < 0)
                return -days;
            return days;
        }
};
int main() {
    Date d1(10,7,2016), d2(10,7,1995), d3;
    d1.display();
    cout << "+";
    d2.display();
    cout << "=";
    d3 = d1 + d2;
    d3.display();
    cout << "\n Days between them: " << d1-d2;
    return 0;
}
```

5. **WAP to define a class Distance with necessay data members and functions. Then overload relational operators to compare two objects of distance class.**

```
#include <iostream>
#include <cmath>
using namespace std;
class Distance {
    float kilometre,metre;
    public:
        Distance(float km = 0, float m=0):
kilometre(km), metre(m) {}
        float Total() {
            float m = kilometre*1000 + metre;
            return m;
        }
        void input() {
            cout << "Enter the distance in kilometre
and metre" << endl;
            cin >> kilometre >> metre;
        }
        void display() {
            cout << "(" << kilometre << "," << metre <<
")";
        }
        int operator==(Distance d2) {
            if(Total()==d2.Total())
                return 1;
            return 0;
        }
        int operator<(Distance d2) {
            if(Total()<d2.Total())
                return 1;
            return 0;
        }
        int operator>(Distance d2) {
            if(Total()>d2.Total())
                return 1;
            return 0;
        }
};
```

```
int main() {
    Distance d1,d2;
    d1.input();
    d2.input();
    if(d1==d2) {
        cout << "They are equal" << endl;
    } else if (d1>d2) {
        d1.display();
        cout << " is greater than";
        d2.display();
        cout << endl;
    } else if(d1<d2) {
        d1.display();
        cout << " is less than";
        d2.display();
        cout << endl;
    }
    return 0;
}
```

6. **WAP having a class to represent money. The class should have two integer members to represent rupees and paisa. Overload + and - for adding and subtracting the objects. Then overload >, < , == and != operator for comparing objects.**

```
#include <iostream>
#include <cmath>
using namespace std;
class Money {
    float rupee,paisa;
    public:
        Money(float r = 0, float p=0): rupee(r),
paisa(p) {}
        float Total() {
            float t = rupee*100 + paisa;
            return t;
        }
       void input() {
            cout << "Enter the Money in rupee and
paisa" << endl;
            cin >> rupee >> paisa;
```

```
        }
        void display() {
            cout << rupee << " rupee and " << paisa <<
" paisa.";
       }
       int operator!=(Money m2) {
           if(Total()!=m2.Total())
               return 1;
            return 0;
        }
        int operator==(Money m2) {
            if(Total()==m2.Total())
                return 1;
            return 0;
        }
       int operator<(Money m2) {
            if(Total()<m2.Total())
                return 1;
            return 0;
       }
        int operator>(Money m2) {
            if(Total()>m2.Total())
                return 1;
            return 0;
        }
        Money operator+(Money m2) {
            int total = Total()+m2.Total();
            return Money(total/100,total%100);
        }
};
int main() {
    Money m1,m2;
    m1.input();
    m2.input();
    if(m1!=m2)
        cout << "They are unequal" << endl;
    if(m1==m2) {
        cout << "They are equal" << endl;
    } else if (m1>m2) {
        m1.display();
```

```
        cout << " is greater than ";
        m2.display();
        cout << endl;
    } else if(m1<m2) {
        m1.display();
        cout << " is less than ";
        m2.display();
        cout << endl;
    }
    cout << "On adding, we get ";
    (m1+m2).display();
    return 0;
}
```

7. **Write a class having an array as member. Overload the index operator([]) to input and display the elements.**

```
#include <iostream>
#include <cstdlib>
using namespace std;
const int size = 4;
class Array {
    int a[size];
    public:
        Array(int b[]) {
            for (int i=0;i<size;i++)
                a[i] = b[i];
        }
        int& operator[](int index) {
            if (index<0 || index >size)
            {
                cout << "Bound exception";
                exit(0);
            }
          return a[index];
      }

};
int main() {
    int b[] = {1,3,5,9};
    Array A(b);
```

```
    cout << "Second element is:" << A[1] << endl;
    A[1] = 1;
    cout << "Second element is:" << A[1] << endl;
    cout << "Fifth element is:" << A[5];
    return 0;
}
```

Lab 8

1. **WAP to add two matrices by overloading + operator.**

```
#include <iostream>
using namespace std;
const int ROW = 3;
const int COL = 3;
class Matrix {
int arr[ROW][COL];
    public:
        void input() {
            cout << "Enter the elements of the Matrix:
" << endl;
           for (int i=0;i<ROW;i++) {
                for(int j=0;j<COL;j++) {
                    cout << "a(" << i+1 << "," << j+1
<< "):";
                    cin >> arr[i][j];
                }
            }
        }
        void display() {
            for (int i=0;i<ROW;i++) {
                cout << "|";
                for(int j=0;j<COL;j++) {
                    cout << arr[i][j] << " ";
                }
                cout << "|\n";
            }
        }
        Matrix operator+(Matrix b) {
            Matrix temp;
            for(int i=0;i<ROW;i++) {
```

```
                for(int j=0;j<COL;j++)
                    temp.arr[i][j] = arr[i][j]+
b.arr[i][j];
            }
            return temp;
        }
};
int main() {
    Matrix m1,m2,m3;
    m1.input();
    m2.input();
    m1.display();
    cout << "+" << endl;
    m2.display();
    cout << "=" << endl;
    m3 = m1 + m2;
    m3.display();
    return 0;
}
```

2. **WAP to achieve operation like c2 = 2 + c1, where c1 and c2 are complex numbers. Also for c2 = c1 + 2 using operator overloading.**

```
#include <iostream>
using namespace std;
class complex {
    float real,imag;
    public:
        complex():real(0),imag(0) {}
        complex(float x, float y): real(x),imag(y) {}
        void display() {
            cout << "(" << real << "," << imag << ")";
        }
        friend complex operator+(complex,complex);
        friend complex operator+(int,complex);
};
complex operator+(complex a, complex b) {
    complex temp;
    temp.real = a.real+b.real;
     temp.imag = a.imag+b.imag;
    return temp;
```

```
}
complex operator+(int n, complex b) {
    complex temp;
    temp.real = n+b.real;
    temp.imag = n+b.imag;
    return temp;
}
int main() {
    complex c1(1,2),c2(1,2),c3;
    c3 = c1 + c2;
    c1.display();
    cout << "+";
    c2.display();
    cout << "=";
    c3.display();
    cout << endl;
    cout << "2 +";
    c1.display();
    cout << " = ";
    c3 = 2 + c1;
    c3.display();
    return 0;
}
```

3. **WAP that shows the conversion from basic type to class type.**

```
#include <iostream>
using namespace std;
class Time {
    int h,m,s;
    public:
        Time():h(0),m(0),s(0) {}
        Time(int s) {
            m = s/60;
            s %= 60;
            h = m/60;
            m %= 60;
        }
        void display() {
            cout << h << ":" << m << ":" << s << endl;
        }
```

```
};
int main() {
    Time t;
    int duration = 90;
    cout << duration << " seconds is:" << endl;
    t = duration;
    t.display();
    return 0;
}
```

4. **WAP that shows the conversion from class type to basic type.**

```
#include <iostream>
using namespace std;
class Number {
    int n;
    public:
        void display() {
            cout << n;
        }
        Number(int x): n(x) {}
        operator int() {
            return n;
        }
};
int main() {
    Number n(5);
    int a = n;
    cout << a;
    return 0;
}
```

5. **WAP to convert object of a class that represents weight of gold in tola to object of class that represents weight in grams. (1 tola = 1.664 gm)**

```
#include <iostream>
using namespace std;
class tola {
    float tol;
    public:
        tola(float t=0.0): tol(t) {}
```

```
        float gettola() {
            return tol;
        }
        void display() {
            cout << "Weight in tola: " << tol << endl;
        }
};
class grams {
    float gms;
    public:
        grams(float g = 0.0):gms(g) {}
        grams(tola t) {
            gms = 1.664 * t.gettola();
        }
        void display() {
            cout << "Weiht in grams: " << gms << endl;
        }
};
int main() {
    tola t(1);
    t.display();
    grams g;
    g = t;
    g.display();
    return 0;
}
```

6. **WAP to convert type of object of polar class to rectangle class and vice versa. Apply constructor functions in both the class.**

```
#include <iostream>
#include <cmath>
using namespace std;
class rectangle;
class polar {
    float r,theta;
    public:
        polar(float p=0,float q=0): r(p), theta(q) {}
        polar(rectangle);
        float get_r() {
            return r;
```

```
        }
        float get_theta() {
            return theta;
        }
        void display() {
            cout << "Polar: (" << r << ", " << theta <<
")" << endl;
        }
};
class rectangle {
    float x,y;
    public:
        rectangle(float p=0, float q=0): x(p), y(q) {}
        rectangle(polar p) {
            float r = p.get_r();
            float theta = p.get_theta();
            x = r * cos(theta);
            y = r * sin(theta);
        }
        float get_x() {
            return x;
        }
        float get_y() {
            return y;
        }
        void display() {
            cout << "Rectangle: (" << x << "," << y <<
")" << endl;
        }
};
polar::polar(rectangle r) {
    float x = r.get_x();
    float y = r.get_y();
    r = sqrt(x*x+y*y);
    theta = atan(y/x);
}
int main() {
    rectangle r1(1,2), r2;
    polar p1(2,30),p2;
    r2 = p1;
```

```
    p2 = r1;
    p1.display();
    r2.display();
    r1.display();
    p2.display();
    return 0;
}
```

Lab 9

1. **WAP that will inherit a class where member are increased by its member function. The derived class should override the function that increase its member in the derived class that will add increment of the derived class data member. Create other meaningful function that suit your requirement.**

```
#include <iostream>
using namespace std;
class Number {
    protected:
       int n;
    public:
        Number():n(0) {}
        void increase() {
            n++;
        }
};
class Counter:public Number {
    int count;
    public:
        Counter(): count(0) {}
        void increase() {
            Number::increase();
            count++;
        }
        void display() {
            cout << "Count of base class = " << n <<
endl;
```

```
            cout << "Count of derived class = " <<
count << endl;
        }
};
int main() {
    Counter c;
    c.increase();
    c.display();
    c.increase();
    c.display();
    return 0;
}
```

2. **WAP to show ambiguity and its resolution in multiple inheritance.**

```
#include <iostream>
using namespace std;
class Number {
    protected:
    int n;
   public:
       Number():n(0) {}
        void increase() {
            n++;
        }
};
class Counter:public Number {
    int count;
    public:
        Counter(): count(0) {}
        void increase() {
            Number::increase();
            count++;
        }
        void display() {
            cout << "Count of base class = " << n <<
endl;
            cout << "Count of derived class = " <<
count << endl;
        }
};
```

```
int main() {
    Counter c;
    c.increase();
    c.display();
    c.increase();
    c.display();
    return 0;
}
```

3. **WAP with three classes student, test and result by using multilevel inheritance. Assume necessary data members and functions yourself and program with input information, input data and calculate marks, total and display the result.**

```
#include <iostream>
using namespace std;
class Number {
    protected:
        int n;
    public:
        Number():n(0) {}
        void increase() {
            n++;
        }
};
class Counter:public Number {
    int count;
    public:
        Counter(): count(0) {}
        void increase() {
            Number::increase();
            count++;
        }
        void display() {
            cout << "Count of base class = " << n <<
endl;
            cout << "Count of derived class = " <<
count << endl;
        }
};
int main() {
```

```
    Counter c;
    c.increase();
    c.display();
    c.increase();
    c.display();
     return 0;
}
```

4. **WAP with a class cricketer that has data member to repesent name, age and number of match played. From this class cricketer, derive two classes, bowler and batsman. The bowler class should have number of wicket as data member and batsman class should have number of runs and number of centuries as data members. Use appropriate member functions in all classes to make your program meaningful.**

```
#include <iostream>
using namespace std;
class Number {
    protected:
      int n;
    public:
        Number():n(0) {}
        void increase() {
            n++;
        }
};
class Counter:public Number {
    int count;
    public:
        Counter(): count(0) {}
        void increase() {
            Number::increase();
            count++;
        }
        void display() {
            cout << "Count of base class = " << n <<
endl;
            cout << "Count of derived class = " <<
count << endl;
```

```
        }
};
int main() {
    Counter c;
    c.increase();
    c.display();
    c.increase();
    c.display();
    return 0;
}
```

5. **WAP that shows the concept of hybrid inheritance.**

```
#include <iostream>
using namespace std;
class Number {
    protected:
        int n;
    public:
        Number():n(0) {}
        void increase() {
            n++;
        }
};
class Counter:public Number {
    int count;
    public:
        Counter(): count(0) {}
        void increase() {
           Number::increase();
            count++;
        }
        void display() {
            cout << "Count of base class = " << n <<
endl;
            cout << "Count of derived class = " <<
count << endl;
        }
};
int main() {
    Counter c;
    c.increase();
```

```
    c.display();
    c.increase();
    c.display();
    return 0;
}
```

6. **Define a class publication which has a title. Derive two classes from it. A class book which has an accession number and a class magazine which has volume number. With these two as bases, derive the class journal. Define a function print() in each of these classes. Ensure that the derived class function always invokes the base class function. In main, create a journal called IEEOOP with an accession number 681.3 and a volume number 1. Invoke the print() function for this object.**

```
#include <iostream>
using namespace std;
class Number {
    protected:
        int n;
    public:
       Number():n(0) {}
       void increase() {
            n++;
        }
};
class Counter:public Number {
    int count;
    public:
        Counter(): count(0) {}
        void increase() {
            Number::increase();
            count++;
        }
        void display() {
            cout << "Count of base class = " << n <<
endl;
            cout << "Count of derived class = " <<
count << endl;
```

```
        }
};
int main() {
    Counter c;
    c.increase();
    c.display();
    c.increase();
    c.display();
    return 0;
}
```

Lab 10

1. **WAP that shows runtime polymorphism in C++.**

```
#include <iostream>
using namespace std;
class Base {
    public:
        virtual void display() {
            cout << "Called from base class." << endl;
        }
};
class Derived1: public Base {
   public:
       void display() {
            cout << "Called from derived class 1." <<
endl;
        }
};
class Derived2: public Base {
    public:
        void display() {
           cout << "Called from derived class 2." <<
endl;
        }
};
int main() {
    Base *bp;
    bp = new Base();
    bp->display();
    bp = new Derived1();
```

```
    bp->display();
    bp = new Derived2();
    bp->display();
    return 0;
}
```

2. **WAP having student as an abstract class and create derived class such as engineering, science and medical. Create their objects, process them and access them using base class pointer.**

```
#include <iostream>
using namespace std;
class student {
    protected:
        int roll;
        char name[25];
    public:
        virtual void display() = 0;
        virtual void input() = 0;
};
class engineering: public student {
    int back_exams;
    public:
        void input() {
            cout << "Enter name:";
            cin >> name;
            cout << "Enter roll number:";
            cin >> roll;
            cout << "Number of back exams:";
            cin >> back_exams;
        }
        void display() {
            cout << "Name: " << name << endl;
            cout << "Roll number: " << roll << endl;
            cout << "Number of backs: " << back_exams
<< endl;
        }
};
class science: public student {
```

```
    float percent;
    public:
        void input() {
            cout << "Enter name:";
            cin >> name;
            cout << "Enter roll number:";
            cin >> roll;
            cout << "Percentage: ";
            cin >> percent;
        }
        void display() {
            cout << "Name: " << name << endl
                 << "Roll no:" << roll << endl
                 << "Percentage:" << percent << endl;
       }
};
class medical:public student {
    int num_operations;
    public:
        void input() {
            cout << "Enter name:";
            cin >> name;
            cout << "Enter roll number:";
            cin >> roll;
            cout << "Number of operations:";
           cin >> num_operations;
        }
        void display() {
            cout << "Name: " << name << endl
                 << "Roll no:" << roll << endl
                 << "Number of operations: " <<
num_operations << endl;
       }
};
int main() {
    engineering e;
    science s;
    medical m;
    student *sp[] = {&e,&s,&m};
    for (int i=0;i<3;i++) {
```

```
        sp[i]->input();
        sp[i]->display();
    }
    return 0;
}
```

3. **WAP that shows the use of pure virtual function.**

```
#include <iostream>
using namespace std;
class Base {
   public:
       virtual void display()=0;
};
class Derived1: public Base {
    public:
        void display() {
            cout << "Called from derived class 1." <<
endl;
        }
};
class Derived2: public Base {
    public:
        void display() {
            cout << "Called from derived class 2." <<
endl;
        }
};
int main() {
    Base *bp;
    bp = new Derived1();
    bp ->display();
    bp = new Derived2():
    bp->display();
    return 0;
}
```

4. **A base class pointer contains address of its derived class object and if base class pointer goes out of scope, which destructor is called. WAP to illustrate this.**

```
#include <iostream>
```

```
using namespace std;
class Base {
   public:
        virtual ~Base() {
            cout << "Called base class destructor" <<
endl;
        }
};
class Derived: public Base {
    public:
        ~Derived() {
            cout << "Called derived class destructor"
<< endl;
}
};
int main() {
    Base *bp;
    bp = new Derived();
    delete bp;
    return 0;
}
```

5. **WAP that shows RTTI mechanism in C++. a) dynamic_cast b) typeid**

```
#include <iostream>
using namespace std;
class Animal {
    public:
        virtual void display() {
           cout << "Animal class" << endl;
        }
};
class Human: public Animal {
    public:
        void display() {
            cout << "Human class" << endl;
        }
};
class Cow: public Animal {
    public:
        void display() {
```

```
            cout << "Cow class" << endl;
        }
};
void check(Animal *A) {
    Human *H;
    Cow *C;
    if(H=dynamic_cast<Human *>(A))
    {
        cout << "Pointing to human class" << endl;
        A->display();
    } else if((C = dynamic_cast<Cow *>(A))) {
            cout << "Pointing to Cow Class" << endl;
            A->display();
    } else {
        cout << "Pointing to Animal class." << endl;
        A->display();
    }
};
int main() {
    check(new Human());
    check(new Animal());
    check(new Cow());
    return 0;
}

#include <iostream>
#include <typeinfo>
using namespace std;
class Animal {
    public:
        virtual void display() {
            cout << "Animal class" << endl;
        }
};
class Human: public Animal {
    public:
        void display() {
            cout << "Human class" << endl;
        }
};
```

```
class Cow: public Animal {
    public:
        void display() {
            cout << "Cow class" << endl;
        }
};
void check(Animal *A) {
    cout << "Animal pointer pointing to class: " <<
typeid(*A).name() << endl;
};
int main() {
    check(new Human());
    check(new Animal());
    check(new Cow());
    return 0;
}
```

Lab 11

1. **WAP with class template to represent array and add member functions to find minimum, maximum and sort the generic array.**

```
#include<iostream>
using namespace std;
const int SIZE = 5;
template <class t>
class array{
    t a[SIZE];
    public:
    void input(){
        for(int i=0;i<SIZE;i++) {
            cout << "Element " << i+1 << ": ";
            cin>>a[i];
            }
    }
    void sort(){
    t m=a[0];
        for(int i=0;i<SIZE;i++){
        for(int j=0;j<SIZE;j++){
            if (a[i]<a[j]){
```

```
                    m=a[i];
                    a[i]=a[j];
                    a[j]=m;
                }
            }
        }
        }
        t minimum(){
            return a[0];
        }
        t maximum(){
        return a[4];
        }
};
int main(){
        array<int> x ;
        cout << "Enter the elements of array:" << endl;
        x.input();
        x.sort();
        cout<<"Min = "<<x.minimum() << endl;
        cout<<"Max = "<<x.maximum() << endl;
        array<float> y ;
        y.input();
        y.sort();
        cout<<"Min = "<<y.minimum() << endl;
        cout<<"Max = "<<y.maximum() << endl;
}
```

2. **WAP for finding the sum and average of an array elements using function template.**

```
#include<iostream>
using namespace std;
const int SIZE=5;
template <class t>
void sumAndAverage(t * a){
    t sum = 0;
    for(int i = 0;i < SIZE;i++){
        sum += a[i];
    }
    cout << "Sum = " << sum << "\t average = " << sum /
SIZE << endl;
```

```
}
int main(){
    int t[] = {2,4,5,6,78};
    sumAndAverage<int>(t);
    float g[] = {2.2,4.4,22.1,44.3,2.1};
    sumAndAverage<float> (g);
    return 0;
}
```

3. **Write a function template for the function power() which has two parameters base and exp and returns base^exp. the type of base is the parameter to the template and type of exp is int. If the exp is negative then it must be converted to its positive equivalent.**

```
#include<iostream>
#include<cmath>
using namespace std;
template<class t>
float power(t base, t exp) {
    if(exp<0)
        exp = -exp;
    return pow(base,exp);
}
int main(){
  int a, b;
    cout<<"Value of base:"; cin>>a;
    cout<<"Value of exponential: ";cin>>b;
    cout<<power<int>(a,b);
    float t1,t2;
    cout<<endl<<"Float value of base: ";cin>> t1;
    cout<<"Float value of exponential: ";cin>> t2;
    cout<<power(t1,t2);
    return 0;
}
```

4. **Create a class template with necessary member variables to represent vectors. Write initialization mechanism and overload * operator to find the scalar product of the two vectors. Also write a member function to display the vector. Use**

above template and write main funciton to perform scalar product of vectors of type int and double.

```
#include<iostream>
using namespace std;
const int size=3;
template<class T>
class vect
{
    T* v;
public:
    vect()
    {
        v = new T[size];
        for(int i=0;i<size;i++)
        {
            v[i]=0;
        }
    }
    vect(T* a)
    {
        v = new T[size];
        for(int i=0;i<size;i++)
        {
            v[i]=a[i];
        }
    }
    T operator*(vect &y)
    {
        T sum = 0;
        for(int i=0;i<size;i++)
        {
            sum += v[i]*y.v[i];
        }
       return sum;
  }
};
int main()
{
    int x[3]={1,2,3};
    int y[3]={2,3,4};
```

```
    vect <int>v1(x);
    vect <int>v2(y);
    int product = v1*v2;
    cout<< "Product = "<<product<<endl;
    double p[3]={1.8,2.6,3.4};
    double q[3]={2.8,3.2,4.7};
    vect <double>v3(p);
    vect <double>v4(q);
    double product2 = v3*v4;
    cout<< "Product = "<<product2<<endl;
    return 0;
}
```

5. Define a class to represent time. It should have memvber function to read and display the time. The function to read time should raise an exception if the user enter invalid values for hour, minutes and seconds. The exception thrown should contain an argument. The exception thrown should be thandled outside of the member functions of the class.

```
#include<iostream>
using namespace std;
class Clock {
  int hour,min,sec;
    public:
    void input(){
         cout<<"Hour: ";cin>>hour;
         if(hour>24||hour<0)
                  throw Exception('H');
             cout<<"Min: ";cin>>min;
             if(min>60 or min<0){
                  throw Exception('M');
             }
             cout<<"Sec: ";cin>>sec;
             if(sec>60 or sec<0){
                  throw Exception('S');
             }
    }
    void display(){
         cout<<"Time: " << hour<<" : "<<min<<" : "<<sec;
```

```
    }
    class Exception {
        public:
            char message;
            Exception(char m): message(m) {}
    };
};
int main() {
    Clock t;
    try {
        t.input();
        t.display();
    }
    catch (Clock::Exception err){
         switch(err.message) {
             case 'H': cout << "Hour can't be greater
than 24." << endl;
break;
           case 'M': cout << "Minute can't be greater
than 60." << endl;
 break;
   case 'S': cout << "Second can't be greater than 60."
<< endl;
break;
}
}
    return 0;
}
```

6. **Write a meaningful program that can handle multiple exceptions.**

```
#include<iostream>
using namespace std;
void test(int b){
    if(b == 0)
           throw 1;
        if(b == 1)
            throw 1.0;
        if(b == 2)
            throw 'e';
        cout << "Value of b is: " << b << endl;
```

```
}
int main(){
    try{
        test(3);
        test(4);
        test(0);
        test(1);
    }
    catch(int c) {
         cout<<"Integer exception";
    }
    catch(double a) {
         cout<<"Double exception";
    }
    catch(char e) {
    cout<<"Character exception";
    }
    return 0;
}
```

7. **Write a meaningful program to use both exception specification for function.**

```
#include<iostream>
using namespace std;
void test(int b) throw (int,double){
    if(b==0)
            throw b;
        if(b==1)
            throw 1.0;
        // throw 'r'; //error
}
int main(){
    try {
        test(0);
        test(1);
    }
    catch(int c){
         cout<<"c = "<<c;
    }
```

```
    catch(double a){
        cout<<"a = "<<a;
    }
    return 0;
}
```