6. Write a program to illustrate the concept of derived class constructor in inheritance.

```
#include <iostream>
using namespace std;

class A
{
public:
    A(int nValue)
    {
        cout << "A: " << nValue << endl;
    }
};

class B: public A
{
public:
    B(int nValue, double dValue)
    : A(nValue)
    {
        cout << "B: " << dValue << endl;
    }
};

class C: public B
{
public:
    C(int nValue, double dValue, char chValue)
    : B(nValue, dValue)
    {
        cout << "C: " << chValue << endl;
    }
};

int main()
{
    C cClass(5, 4.3, 'R');

    return 0;
}
```

7a.Subscript operator overloading

```
#include <iostream>
```

```
using namespace std;
const int SIZE = 10;

class safearay
{
  private:
    int arr[SIZE];
  public:
    safearay()
    {
      register int i;
      for(i = 0; i < SIZE; i++)
      {
        arr[i] = i;
      }
    }
    int &operator[](int i)
    {
       if( i > SIZE )
       {
           cout << "Index out of bounds" <<endl;
           // return first element.
           return arr[0];
       }
       return arr[i];
    }
};
int main()
{
  safearay A;
```

```
  cout << "Value of A[2] : " << A[2] <<endl;
  cout << "Value of A[5] : " << A[5]<<endl;
  cout << "Value of A[12] : " << A[12]<<endl;

  return 0;
}
```

7b.+ and== overloading

```
const int sz=80;
enum boolean{false,true};
class string
{
char str[80];
public:
string(){strcpy(str," ");}
string(char s[ ]) {strcpy(str,s);}
void display() {cout << str;}
void getstr() { gets(str);}
boolean operator==(string ss)
{
return
(strcmp(str,ss.str)==0)?true:false;
}
string operator +(string ss)
{
return strcat(str,ss.str);
}
};
void main() {
string s1,s2,s3;
int ch;
cout << "enter the first string\n";
s1.getstr();
cout << "enter second string\n";
s2.getstr();
do { cout << "Menu";
cout << "1.compare two strings\n";
cout << "2.concatinate two strings\n";
cout << "enter your choice\n";
```

```
cin >> ch;
switch(ch) {
case 1:if(s1==s2)
      cout << "strings are equal\n";
      else
      cout << "strings are not equal\n";
      break;
case 2:s3=s1+s2;
      cout << "concatenated string is\n";
      s3.display();
      break;   }
}while(ch==1);
getch(); }
```

6.Write a C++ program to create a class called STRING and implement the following operations.

Display the results after every operation by overloading <<.

i) STRING s1 = “ISE”

ii) STRING s2 = “MSRIT”

iii) STRING s3 = s1+s2 (Use copy constructor)

```
#include<iostream>

#include<cstring>

#include<cstdlib>

using namespace std;

class strng

{

char str[20];

public:

strng()
```

```
{
str[0]=&#39;\0&#39;;
}
strng(char temp[])
{
strcpy(str,temp);
}

strng(strng &temp)
{
strcpy(str,temp.str);
}
void display();
friend strng operator+(strng s1,strng s2);
friend ostream & operator<<(ostream&,strng&);
};
void strng::display()
{
cout<<"\nstring is"<<str;
}
strng operator+(strng s1,strng s2)
{
strcat(s1.str,s2.str);
```

```
return s1;
}
ostream & operator<<(ostream& os,strng& s)
{
os<<s.str<<endl;
return os;
}
int main()

{
strng s1("ise");
strng s2("MSRIT");
strng s3;
cout<<"\nBEFORE CONCATINATION";
cout<<"\n s1="<<s1;
cout<<"\n s2="<<s2;
s3=s1+s2;
cout<<"\nAFTER CONCATENATION"<<"\n:";
cout<<"\n s1+s2="<<s3;
return 0;
}
```

7d.

Write a C++ program to create a class called OCTAL, which has the characteristics of an octal number. Implement the following operations by writing an appropriate constructor and an overloaded operator +.

i. OCTAL h = x ; where x is an integer

ii. int y = h + k ; where h is an OCTAL object and k is an integer.

Display the OCTAL result by overloading the operator <<. Also display the values of h and y.

```
#include<iostream>
using namespace std;
class octal
{
int oct,dec,ten;
public:
octal()
{
oct=0;
ten=1;
}
void operator=(int x)
{
int r;
dec=x;
while(x!=0)
{
```

```
r=x%8;
x=x/8;
oct=oct+ten*r;
ten=ten*10;
}
}
int operator+(int k)
{
return(dec+k);
}
friend ostream & operator<<(ostream&,octal& c);
};
ostream & operator<<(ostream& sout,octal&c)
{
sout<<c.oct;
}
int main()
{
octal h;
int n,k;
cout<<"Enter a integer to change to octal: ";
cin>>n;
cout<<endl;
```

```
h=n;

cout<<"The octal value of "<<n<<" is: "<<h<<endl;

cout<<"\nEnter integer to be added to previous octal: ";

cin>>k;

cout<<endl;

int y=h+k;

cout<<"Integer sum of octal and integer is: "<<y<<"\n";

return 0;

}
```

9. Write a C++ Program to implement Type Conversion

```
#include <iostream>
using namespace std;
const float MeterToFloat=3.280833;
class Distance {
	int feets;
	float inches;
	public:
	Distance()  //Distance Constructor {
		feets=0;
		inches=0.0;
	}
	Distance(float numofmeters)  //Single Parameter constructor {
		float feetsinfloat= MeterToFloat * numofmeters;
		feets=int(feetsinfloat);
		inches=12*(feetsinfloat-feets);
	}
	void displaydist()  // Method to display converted values {
		cout<<"Converted Value is: "<<feets<<"\' feets and "<<inches<<'\"'<<" inches.";
	}
};
int main() {
	cout <<"Float to distance conversion.\n*********************************\n";
```

```
    float meters;
    cout<<"Enter values in meter:";
    cin >>meters;
    Distance distance = meters;
    distance.displaydist();
}
```

10.Write a C++ Program to implement virtual functions and dynamic polymorphism

```
#include<iostream>
using namespace std;


class A
{
   public:
   virtual void show()
   {
      cout << "Base class\n";
   }
};


class B: public A
{
private:
   virtual void show()
   {
      cout << "Derived class\n";
   }
};
```

```
int main()
{
   A *a;
   B b;
   a = &b;
   a -> show();
```

```
}
```

11. Write a C++ program to illustrate the File handling

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
#include<fstream.h>
void main()
{
char c,fname[10];
ofstream out;
cout<<"Enter File name:";
cin>>fname;
out.open(fname);
cout<<"Enter contents to store in file (Enter # at end):\n";
while((c=getchar())!='#')
{
out<<c;
}
out.close();
getch();
}
```

12. a. Write C++ program to use try catch statements to handle division by zero and out-of-

bounds exception.

```
#include<iostream>

using namespace std;
```

```
int main()
{
int a,b;
cout<<"enter a,b";
cin>>a>>b;
try
{
if(b!=0)
{
cout<<"result is:"<<(a/b);
}
else
throw(b);
}
catch(int)
{
cout<<"division not possible";
}
return 0;
}
```

13.Write a C++ program to create a template function for Bubble Sort and demonstrate sorting of

integers and doubles

```
#include<iostream>

using namespace std;

template<class t>

void bubble(t a[],int n)

{

for(int i=0;i<n-1;i++)

for(int j=n-1;i<j;j- -)

if(a[j]<a[j-1])

{

swap(a[j],a[j-1]);

}

}

template<class x>

void swap(x &a,x &b)

{

x temp=a;

a=b;

b=temp;

}

int main()

{

int x[5]={10,50,30,40,20};

float y[5]={1.1,5.5,3.3,4.4,2.2};
```

```
bubble(x,5);

bubble(y,5);

cout<<"sorted x arrays";

for(int i=0;i<5;i++)

cout<<x[i]<<" ";

cout<<"\n";

cout<<"sorted y arrays";

for(int j=0;j<5;j++)

cout<<y[j]<<" ";

cout<<"\n";

return 0;

}
```

Overload

8.

```
#ifndef POINT_H
#define POINT_H
#include <iostream>

class Point {
private:
  int x, y;

public:
  explicit Point(int x = 0, int y = 0);
  int getX() const;
  int getY() const;
  void setX(int x);
  void setY(int y);
  Point & operator++();              // ++prefix
  const Point operator++(int dummy); // postfix++
  const Point operator+(const Point & rhs) const; // Point + Point
  const Point operator+(int value) const;          // Point + int
```

```
   Point & operator+=(int value);           // Point += int
   Point & operator+=(const Point & rhs); // Point += Point

   friend std::ostream & operator<<(std::ostream & out, const Point & point); // out << point
   friend std::istream & operator>>(std::istream & in, Point & point);        // in >> point
   friend const Point operator+(int value, const Point & rhs); // int + Point
};

#include "Point.h"
#include <iostream>
using namespace std;

// Constructor - The default values are specified in the declaration
Point::Point(int x, int y) : x(x), y(y) { }

// Getters
int Point::getX() const { return x; }
int Point::getY() const { return y; }

// Setters
void Point::setX(int x) { this->x = x; }
void Point::setY(int y) { this->y = y; }

// Overload ++Prefix, increase x, y by 1
Point & Point::operator++() {
   ++x;
   ++y;
   return *this;
}

const Point Point::operator++(int dummy) {
   Point old(*this);
   ++x;
   ++y;
   return old;
}

// Overload Point + int. Return a new Point by value
const Point Point::operator+(int value) const {
   return Point(x + value, y + value);
}

// Overload Point + Point. Return a new Point by value
const Point Point::operator+(const Point & rhs) const {
   return Point(x + rhs.x, y + rhs.y);
}
```

```
// Overload Point += int. Increase x, y by value
Point & Point::operator+=(int value) {
   x += value;
   y += value;
   return *this;
}

// Overload Point += Point. Increase x, y by rhs
Point & Point::operator+=(const Point & rhs) {
   x += rhs.x;
   y += rhs.y;
   return *this;
}

// Overload << stream insertion operator
ostream & operator<<(ostream & out, const Point & point) {
   out << "(" << point.x << "," << point.y << ")";
   return out;
}

// Overload >> stream extraction operator
istream & operator>>(istream & in, Point & point) {
   cout << "Enter x and y coord: ";
   in >> point.x >> point.y;
   return in;
}

// Overload int + Point. Return a new point
const Point operator+(int value, const Point & rhs) {
   return rhs + value;  // use member function defined above
}

#include <iostream>
#include "Point.h"
using namespace std;

int main() {
   Point p1(1, 2);
   cout << p1 << endl;   // (1,2)

   Point p2(3,4);
   cout << p1 + p2 << endl; // (4,6)
   cout << p1 + 10 << endl; // (11,12)
   cout << 20 + p1 << endl; // (21,22)
   cout << 10 + p1 + 20 + p1 << endl; // (32,34)
```

```
    p1 += p2;
    cout << p1 << endl; // (4,6)
    p1 += 3;
    cout << p1 << endl; // (7,9)

    Point p3;  // (0,0)
    cout << p3++ << endl; // (0,0)
    cout << p3 << endl;   // (1,1)
    cout << ++p3 << endl; // (2,2)
}
```