OOP Practical Programs with Solution.

1. Write a program to swap two numbers using reference variable concept.

```
#include<iostream>
 using namespace std;
 void swap(int &x, int &y)
 {
     int temp;
     temp = x;
     x = y;
     y = temp;
 }
 int main()
 {
     int a, b;
     cout<<"Enter the value of a: ";
     cin>>a;
     cout<<"Enter the value of b: ";
     cin>>b;
     cout<<endl<<"Before swapping: ";
     cout<<"a= "<<a<<" and b= "<<b;
     swap(a, b);
     cout<<endl<<"After swapping: ";
     cout<<"a= "<<a<<" and b= "<<b;
 }
```

2. Write an inline function to find maximum of 3 numbers.

```
  #include <iostream>

  using namespace std;

  inline int cmp(int x,int y,int z)
```

```
{
   if(x>y&&x>z)
   return(x);
   else if(y>z)
   return(y);
   else
   return(z);


}
int main()
{
   int a,b,c;
   cout<<"enter three numbers:"<<endl;
   cin>>a>>b>>c;
   cout<<cmp(a,b,c)<<" is larger"<<endl;
   return 0;
  }
```

3. Write a function called power that takes a double value for n and an int value for p, and returns the result as a double value. Use a default argument of 2 for p, so that if this argument is omitted the number n will be squared.

```
#include<iostream>
using namespace std;
double Power(double n, int p)
{
int i;
double t=n;
for(i=1;i<p;i++)
{
    t*=n;
}
return t;
}
double Power(double n)
{
return n*n;
}

int main()
{
    double n;
    int p,ch;
    cout << "Enter value: " ;
    cin >> n;
    cout << "Enter 1 to enter power or 0 to use default power: ";
    cin >> ch;
    if(ch==0)
    {
     cout <<double(Power(n));
     return 0;
    }
    cout << "Enter power: ";
    cin >> p;
    cout << double(Power(n, p));
    return 0;

}
```

4. **Write a function that takes two Distance values as arguments and returns the larger. Include a main() program that accepts two Distance values from the user, compare them, and displays the larger.(use object as function argument, Write the function inside the class)**

```
#include <iostream>
using namespace std;
struct Distance
{
    int feet;
    float inch;
}d1 , d2, sum;
int main()
{
    cout << "Enter 1st distance," << endl;
    cout << "Enter feet: ";
    cin >> d1.feet;
    cout << "Enter inch: ";
    cin >> d1.inch;
    cout << "\nEnter information for 2nd distance" << endl;
    cout << "Enter feet: ";
    cin >> d2.feet;
    cout << "Enter inch: ";
    cin >> d2.inch;
    sum.feet = d1.feet+d2.feet;
    sum.inch = d1.inch+d2.inch;
    // changing to feet if inch is greater than 12
    if(sum.inch > 12) {
        // extra feet
        int extra = sum.inch / 12;
        sum.feet += extra;
        sum.inch -= (extra * 12);
    }
    cout << endl << "Sum of distances = " << sum.feet << " feet  " << sum.inch << " 
inches";
    return 0;
}
```

5. Write a C++ program to solve Quadratic Equation using constructor.

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

    float a, b, c, x1, x2, discriminant, realPart, imaginaryPart;
    cout << "Enter coefficients a, b and c: ";
    cin >> a >> b >> c;
    discriminant = b*b - 4*a*c;

    if (discriminant > 0)
        {
        x1 = (-b + sqrt(discriminant)) / (2*a);
        x2 = (-b - sqrt(discriminant)) / (2*a);
        cout << "Roots are real and different." << endl;
        cout << "x1 = " << x1 << endl;
        cout << "x2 = " << x2 << endl;
    }

    else if (discriminant == 0)
        {
        cout << "Roots are real and same." << endl;
        x1 = -b/(2*a);
        cout << "x1 = x2 =" << x1 << endl;
    }
    else
        {
```

```
        realPart = -b/(2*a);
        imaginaryPart =sqrt(-discriminant)/(2*a);
        cout << "Roots are complex and different."  << endl;
        cout << "x1 = " << realPart << "+" << imaginaryPart << "i" << endl;
        cout << "x2 = " << realPart << "-" << imaginaryPart << "i" << endl;
    }
    return 0;
}
```

6. Write a C++ program to make arithmetic calculator using inline function.

```
# include <iostream>
using namespace std;
int main()
{
  char op;
  float num1, num2;
  cout << "Enter operator: +, -, *, /: ";
  cin >> op;
  cout << "Enter two operands: ";
  cin >> num1 >> num2;
  switch(op)
{
     case '+':
      cout << num1 << " + " << num2 << " = " << num1 + num2;
      break;

      case '-':
      cout << num1 << " - " << num2 << " = " << num1 - num2;
      break;

    case '*':
      cout << num1 << " * " << num2 << " = " << num1 * num2;
      break;

    case '/':
      cout << num1 << " / " << num2 << " = " << num1 / num2;
      break;
    default:
      // If the operator is other than +, -, * or /, error message is shown
      cout << "Error! operator is not correct";
      break;
  }
  return 0;
}
```

7. Write a program to derive a class rectangle from base class shape using single inheritance.

```
#include<iostream>
using namespace std;
class Shape
{
    public: double a,b;
        void get_data ()
        {
            cin>>a>>b;
        }
        virtual void display_area () = 0;
};

class Triangle:public Shape
{
    public: void display_area ()
    {
        cout<<"Area of triangle "<<0.5*a*b<<endl;
    }
};

class Rectangle:public Shape
```

```
{
   public: void display_area ()
   {
      cout<<"Area of rectangle "<<a*b<<endl;
   }
};

int main()
{
   Triangle t;
   Shape *st = &t;
   cout<<"Enter base and altitude: ";
   st->get_data();
   st->display_area();
   Rectangle r;
   Shape *sr = &r;
   cout<<"Enter length and breadth: ";
   sr->get_data();
   sr->display_area();
   return 0;
}
```

8. Define a class containing operator function to overload unary minus (‘-‘) operator.

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

class Numbers
{
      int x, y, z;
   public:
      void accept()
      {
```

```
                cout<<"\n Enter Three Numbers";
                cout<<"\n --------------------------";
                cout<<"\n First Number   :  ";
                cin>>x;
                cout<<"\n Second Number  :  ";
                cin>>y;
                cout<<"\n Three Number   :  ";
                cin>>z;
                cout<<"\n --------------------------";
        }
        void display()
        {
                cout<<" ";
                cout<<x<<"\t"<<y<<"\t"<<z;
        }
        void operator-()
        {
                x=-x;
                y=-y;
                z=-z;
        }
};
int main()
{
        Numbers num;
        num.accept();
        cout<<"\n Numbers are :\n\n";
        num.display();
        -num;    //Overloaded Unary (-) Operator
        cout<<"\n\n Negated Numbers are :\n\n";
        num.display();
        return 0;
}
```

9. Write a C++ program for Exception Handling Divide by zero Using C++ Programming.

```
#include<iostream>
using namespace std;
int main()
{
int var1,var2;
float var3;
cout<<"enter the dividend:";
cin>>var1;
cout<<"\n";
cout<<"enter the divisor:";
cin>>var2;
cout<<"\n";
//exception handling begins here
try //try block
{
if(var2!=0) //checking if divisor is zero
{
var3=var1/var2;
cout<<"outcome :"<<var3;
}
else
{
throw(var2); //throwing the exception found
}
}
//catch block
catch(int exc)
{
```

```
cout<<"division by zero is not possible. Please try again with different value of
variables";
}
}
```

10. Write a C++ program to sort an array in ascending order using function template.

```
#include <iostream>
using namespace std;
int main(){
    int i, j, size, temp;
    int arr[25];
    // Asking for input
    cout << "Enter the total no. of elements: ";
    cin >> size;
    // Enter the elements
    cout << "Enter the elements of the array: " << endl;
    for (i = 0; i < size; i++){
        cin >> arr[i];
    }
    // Sorting elements in ascending order
    for (i = 0; i < size; i++){
        for (j = i; j < size; j++){
            if (arr[i] > arr[j+1]){
                temp = arr[i];
                arr[i] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
    // Displaying output
    cout << "Elements sorted in the ascending order are: " << endl;
    for (i = 1; i <= size; i++){
     cout << arr[i] << endl;
    }
    return 0;
}
```