1. Define a function to check whether a given number is a Prime number or not.

```
#include<iostream>
using namespace std;
class PrimeNumber
{
public:
   int n;
   void checkPrime()
   {
      int flag=0;
      for(int i=2;i<=n/2;i++)
      {
         if(n % i == 0)
            flag = 1;
         break;
      }
   if(flag ==0 )
   cout<<"The number is prime"<<endl;
   else
      cout<<"The number is not prime"<<endl;
  }
};
int main()
{
   PrimeNumber a;
   cout<<"Enter the number"<<endl;
   cin>>a.n;
   a.checkPrime();
   return 0;
```

```
}
```

2. Define a function to find the highest value digit in a given number.

```
#include<iostream>
using namespace std;
class MaxNum
{
public:
    int n;
    void MaxDigit()
   {
      int max =-1;
      while( n!=0)
   {
      int rem = n%10;
      if(rem>max)
       max = rem;
       n = n/10;
   }
   cout<<"The maximum digit  is "<<max<<endl;
   }
};
int main()
{
    MaxNum a;
   cout<<"Enter the number"<<endl;
   cin>>a.n;
   a.MaxDigit();
   return 0;
}
```

3. Define a function to calculate x raised to the power y.

```
#include<iostream>
using namespace std;
class xPowerY
{ public:
   int x,y;
   void xPower()
{
    int result = 1;
     while(y!=0)
   {
      y--;
      result = result *x;
   }
   cout<<"x power y is "<<result<<endl;
}
};
int main()
 {
   xPowerY a;
  cout<<"Enter two numbers"<<endl;
  cin>>a.x>>a.y;
  a.xPower();
  return 0;
 }
```

4. Define a function to print Pascal Triangle up to N lines.

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

```
{
   int num;
   int coef;
public:
   PascalTriangle(int n)
   {
      num = n;
   }
   void pascaltri()
   {
         for(int i =1;i<=num;i++)
      {
          int coef =1;
          for(int k=num-i;k>0;k--)
      {
             cout<<" ";
      }    for(int j=1;j<=i;j++)
         {
            cout<<coef<<" ";
            coef=coef*(i-j)/j;
         }cout<<endl;
      }
   }
};
int main()
{
   int x;
   cout<<"Enter number of lines"<<endl;
   cin>>x;
```

```
    PascalTriangle p(x);
    p.pascaltri();
    return 0;
}
```

5. Define a function to check whether a given number is a term in a Fibonacci series or not.

```
#include<iostream>
using namespace std;
class Fibnacci
{
    int num,count=2;
public:
    int a=1,b=1,c=0;
    Fibnacci(int n)
    {
        num = n;
    }
      void checking_fibonacci()
 {
       if(num==0 || num==1)
    {
        cout<<"it is a fibnacci number"<<endl;
    }
      else
    {
       a=0;
       b=1;
       c=a+b;
       while(c<num)
        {
```

```
        a=b;
        b=c;
        c=a+b;
        }
   if(num==c)
       {
        cout<<"Number is in fibnacci series"<<endl;
       }
      else
         cout<<"Number is not in fibnacci series"<<endl;
   }
   }
};
int main()
{
   int x;
   cout<<"Enter a number"<<endl;
   cin>>x;
   Fibnacci A(x);
   A.checking_fibonacci();
   return 0;
}
```

6. Define a function to swap data of two int variables using call by reference.

```
#include<iostream>
using namespace std;
class Swap
{
public :
   int p,q;
```

```
   void swap(int &p,int &q)
   {
      int temp = p;
      p = q;
      q = temp;
   }
};
int main()
{
    int x=5,y=10;
    swap(x,y);
    cout<<"After swapping x = "<<x<< " and y = " <<y<<endl;
    return 0;
}
```

7. Write a function using the default argument that is able to add 2 or 3 numbers.

```
#include<iostream>
using namespace std;
   int add(int x,int y,int z=0)
{
   return x+y+z;
}
  int main()
{
    int x,y,z;
    cout<<"Addition is "<<add(2,3)<<endl;
    cout<<"Addition of 3 numbers is "<<add(1,2,3)<<endl;
    return 0;
}
```

8. Define overloaded functions to calculate area of circle, area of rectangle and area of triangle.

```
#include<iostream>
using namespace std;
class Area
{
public:
   void area(int radius)
   {
      cout<<"Area of circle is = "<<3.14*radius*radius<<endl;
   }
   void area(double l,double b)
   {
      cout<<"Area of rectangle is = "<<l*b<<endl;
   }
   void area(int base,int h)
   {
      cout<<"Area of triangle is = "<<0.5*base*h<<endl;
   }
};
int main()
{
   Area a;
   a.area(4);
   a.area(0.5,3.0);
   a.area(5,2);
   return 0;
}
```

9. Write functions using function overloading to find a maximum of two numbers and both the numbers can be integer or real.

```
#include<iostream>
using namespace std;
class maximum
{
   int max;
public:
   findmax_num(int a,int b)
   {
      if(a>b)
      {
         cout<<"Maximum number is "<<a<<endl;
      }
      else
         cout<<"Maximum number is "<<b<<endl;
   }
    findmax_num(double a,double b)
   {
      if(a>b)
      {
         cout<<"Maximum number is "<<a<<endl;
      }
      else
         cout<<"Maximum number is "<<b<<endl;
   }
};
int main()
{
```

```
    maximum x;
    x.findmax_num(76,98);
    x.findmax_num(55.99,56.1);
    return 0;
}
```

10. Write function using function overloading to add two numbers having different data types.

```
#include<iostream>
using namespace std;
class Adding
{  public:
   add(int a,int b)
   {  int c;
      c = a+b;
      cout<<"Addition of "<<a<<" and "<<b<<" is = "<<c<<endl;
   }
   add(double a,double b)
   {   double c;
      c = a+b;
      cout<<"Addition of "<<a<<" and "<<b<<" is = "<<c<<endl;
   } };
int main()
{
    Adding x;
    x.add(25,45);
    x.add(55.5,44.5);
    return 0;
}
```