```
1       // Assignment-3(OOP)
2
3
4
5   #include<iostream>
6   using namespace std;
7   class Complex
8   {
9   private:
10        int real,imag;
11  public:
12   Complex(int r=0, int i=0)
13        {
14         real=r;
15          imag=i;
16       }
17      friend ostream & operator <<
    (ostream &out,Complex
    const&obj);
18      friend istream & operator >>
    (istream &in,Complex  &obj);
19
20       Complex operator*(Complex
    const&obj)
21        {
```

```
22          Complex result;
23
    result.real=real*obj.real;
24
    result.imag=imag*obj.imag;
25          return result;
26        }
27      };
28
29   ostream &operator << (ostream
    &out,Complex const&obj)
30   {
31            out<<obj.real;
32            out<<"+i"<<obj.imag;
33            return out;
34   }
35
36     istream & operator >> (istream
    &in,Complex  &obj)
37        {
38         in>>obj.real;
39         in>>obj.imag;
40          return in;
41         }
42      int main()
```

```
43      {
44        Complex c1,c2,c3;
45
46      cin>>c1;
47      cin>>c2;
48      cout<<c1<<endl;
49      cout<<c2<<endl;
50      c3=c1*c2 ;
51      cout<<c3<<endl;
52      return 0;
53        }
54
55  Output:
56  3+i6
57  4+i3
58  12+i18
59
60
```

```
#include<iostream>
using namespace std;
class Complex
{
private:
      int real,imag;
public:
 Complex(int r=0, int i=0)
       {
        real=r;
         imag=i;
      }
    void print ()
      {

cout<<real<<"+i"<<imag<<endl;
       }
     Complex operator*(Complex
const&obj)
       {
        Complex result;

result.real=real*obj.real;

result.imag=imag*obj.imag;
```

```
23          return result;
24        }
25      };
26      int main()
27      {
28        Complex c1(10,5),c2(2,4);
29        Complex c3=c1*c2;
30        c1.print();
31        c2.print();
32        c3.print();
33        }
34
35  Output:
36  10+i5
37  2+i4
38  20+i20
39
40
```

```cpp
#include <iostream>
using namespace std;
#include<cmath>
 const double PI = 3.14159;
 inline double
sphereVolume( const double r )
 {
 return 4.0 / 3.0 * PI * pow( r,
3 );
 }
 int main()
 {
 double radius;
 cout << "Enter the length of
the radius of your sphere: ";
 cin >> radius;
cout << "Volume of sphere with
radius " << radius << " is " <<
sphereVolume( radius ) << endl;
 return 0;
 }

 Output:
 Enter the length of the radius
of your sphere: 10
```

```cpp
#include <iostream>
using namespace std;
 #include <math.h>
   int main(int argc, char**
argv)
   {
   cout << "pythagorean triples"
<< endl;
 for(int side1 = 1; side1 < 500;
side1 ++)
    {
    for(int side2 = 1; side2 <
500; side2 ++)
 {
    for(int hypotenuse = 1;
hypotenuse < 500; hypotenuse ++)
        {
      if(pow(side1,2) +
pow(side2,2) == pow(hypotenuse,
2) && pow(hypotenuse,2) <= 500)
          {
            cout << side1 << " + "
<< side2 << " = " << hypotenuse
<< endl;
        }
```

```
        }
        }
        }
        return 0;
    }
 Output:
 pythagorean triples
3 + 4 = 5
4 + 3 = 5
5 + 12 = 13
6 + 8 = 10
8 + 6 = 10
8 + 15 = 17
9 + 12 = 15
12 + 5 = 13
12 + 9 = 15
12 + 16 = 20
15 + 8 = 17
16 + 12 = 20
```

```cpp
#include <iostream>
// cin/cout/endl
#include <string> // string
 using namespace std; // Return true if string is palindrome.
bool isPalindrome(string s)
{
if (s.size() <= 1)
   return true;
   else     if (s.at(0) == s.at(s.size() - 1))
   return isPalindrome(s.substr(1, s.size() - 2));
   return false;
}
int main()
{  // An array of teststrings.
string testStrings[] = {  "aaa", "abc", "abba", "radar", "able was i saw elba", "this is not"
   };
   // Check if each string is a palindrome.
   for(string s : testStrings)
```

```
18    cout << boolalpha << "\"" << s
      << "\" is a palindrome: " <<
      isPalindrome(s) << endl;
19
20      return 0;
21  }
22  Output:
23  "aaa" is a palindrome: true
24  "abc" is a palindrome: false
25  "abba" is a palindrome: true
26  "radar" is a palindrome: true
27  "able was i saw elba" is a
      palindrome: true
28  "this is not" is a palindrome:
      false
29
30
```

//A parking garage charges a $2.00 minimum fee to park for up to three hours. The garage charges an additional $0.50 perhour for each hour or part thereof in excess of three hours. The maximum charge for any given 24-hour period is $10.00. Assumethat no car parks for longer than 24 hours at a time. Write a program that will calculate and print the parking charges for each of 3customers who parked their cars in this garage yesterday. You should enter the hours parked for each customer. Your programshould print the results in a neat tabular format and should calculate and print the total of yesterday's receipts. The program shouldduse the function calculateCharges to determine the charge for each customer. Your outputs should appear in

```
the followingformat:
//Car Hours Charge
//1    1.5      2.00
//2    4.0      2.50
//3    24.0     10.00
//TOTAL 29.5 14.50

#include <iostream>
using namespace std;
#include <iomanip>
 using std::setw;
 using std::setprecision;
 using std::setiosflags;

 #include <cmath>
 double
calculateCharges( double );
main()
{
 double hour, currentCharge,
totalCharges = 0.0, totalHours
= 0.0;
int first = 1;

 cout << "Enter the hours
parked for 3 cars: ";
```

```
24    for ( int i = 1; i <= 3; i++ )
      {
25    cin >> hour;
26    totalHours += hour;
27
28    if ( first ) {
29    cout << setw( 5 ) << "Car" <<
      setw( 15 ) << "Hours"
30    << setw( 15 ) << "Charge\n";
31    first = 0; // prevents this
      from printing again
32    }
33
34    totalCharges +=
      ( currentCharge =
      calculateCharges( hour ) );
35    cout <<
      setiosflags( ios::fixed |
      ios::showpoint )
36    << setw( 3 ) << i <<
      setw( 17 ) << setprecision( 1 )
      << hour
37    << setw( 15 ) <<
      setprecision( 2 ) <<
      currentCharge << "\n";
```

```
38    }
39    cout << setw( 7 ) << "TOTAL"
      << setw( 13 ) <<
      setprecision( 1 )
40    << totalHours << setw( 15 ) <<
      setprecision( 2 )
41    << totalCharges << endl;
42    return 0;
43    }
44    double
      calculateCharges( double
      hours )
45    {
46    double charge;
47
48    if ( hours < 3.0 )
49    charge = 2.0;
50    else if ( hours < 19.0 )
51    charge = 2.0 + .5 *
      ceil( hours - 3.0 );
52    else
53    charge = 10.0;
54    return charge;
55    }
```

```
 Output:

  Enter the hours parked for 3
cars: 1 2 3
 Car Hours Charge
 1   1.0    2.00
 2   2.0    2.00
 3   3.0    2.00
TOTAL 6.0   6.00
```

//Q :Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four data members –a part number (type string), a part description (type string), a quantity of the item being purchased (type int) and a price per item (type int). Your class should have a constructor that initializes the four data members. Provide a set and a get function for each data member. In addition, provide a

and a get function for each data member. In addition, provide a member function named get Invoice Amount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as an int value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.Write a test program that demonstrates class Invoice's capabilities

```
2
3   #include<iostream>
4    #include <string>
5    using namespace std;
```

```
6     class Invoice
7     {
8       public:
9       Invoice( string, string,
      int, int );
10     void setPartNumber( string );
      string getPartNumber();
11    void
      setPartDescription(string);
      string getPartDescription();
12    void setItemQuantity(int);
13     int getItemQuantity();
14     void setItemPrice(int);
15     int getItemPrice();
16    int getInvoiceAmount();
17    private:
18     string partNumber;
19      string partDescription;
20       int itemQuantity;
21        int itemPrice;
22     };
23      Invoice::Invoice( string
      number, string description, int
      quantity, int price )
24     {
25       partNumber=number;
```

```
26      partDescription=description;
    if(quantity>0)
    itemQuantity=quantity;
27   else
28    {
29       itemQuantity=0;
    cout<<"Initial quantity was
    invalid."<<endl;
30  }
31   if(price>0)
32    itemPrice=price;
33   else
34   {
35      itemPrice=0;
36   cout<<"Initial price was
    invalid."<<endl;
37  } } void
    Invoice::setPartNumber( string
    number)
38   {
39    if ( number.length() <=25 )
40       partNumber = number;
41     if ( number.length() >25 )
42  {
43  partNumber = number.substr( 0,
    25 );
```

```
44    cout << "Name \"" << number
      <<"\" exceeds maximum length
      (25).\n"<< "Limiting partNumber
      to first 25 characters.\n" <<
      endl;
45    } }
46    void
      Invoice::setPartDescription(str
      ing description )
47    {
48    if ( description.length() <=
      25 )
49    partDescription = description;
      if ( description.length() >
      25 )
50    {
51    partDescription =
52       description.substr( 0, 25 );
      cout << "Name \"" <<
      description <<"\" exceeds
      maximum length (25).\n"<<
      "Limiting partDescription to
      first 25 characters.\n" <<
      endl;
53    } }
54    void
```

```
Invoice::setItemQuantity(int
quantity )
{
 if(quantity>0)
itemQuantity=quantity;
 else
  {
itemQuantity=0;
 cout<<"Initial quantity was
invalid."<<endl;
 } }
 void Invoice::setItemPrice(int
price )
  {
 if(price>0)
  itemPrice=price;
 else
  {
     itemPrice=0;
 cout<<"Initial price was
invalid"<<endl;
 }}
string Invoice::getPartNumber()
{
  return partNumber;
}
```

```
74  string
    Invoice::getPartDescription()
75   {
76     return partDescription;
77     }
78  int Invoice::getItemQuantity()
    {
79    return itemQuantity;
80     }
81   int Invoice::getItemPrice()
82    {
83     return itemPrice;
84     }
85  int Invoice::getInvoiceAmount()
    {
86  return itemQuantity*itemPrice;
87  }
88  int main()
89   {
90   Invoice Invoice1("ed34","Screw
    Guage",2,30);
91  Invoice
    Invoice2("e322","Screws",10,3);
    cout << "Invoice1's initial
    part number is: "<<
    Invoice1.getPartNumber()<<
```

```
"\nand part description is: "<<
Invoice1.getPartDescription()<<
endl;
 cout<< "quantity per item is:
"<<
Invoice1.getItemQuantity()<<
"\nprice per item is: "<<
Invoice1.getItemPrice()<< endl;
  cout<<"Invoice1's total
amount is:"
<<Invoice1.getInvoiceAmount()<<
endl<<endl;
cout << "Invoice2's initial
part number is: "<<
Invoice2.getPartNumber()<<
"\nand part description is: "<<
Invoice2.getPartDescription()<<
endl;
cout<< "quantity per item is:
"<<Invoice2.getItemQuantity()<<
"\nprice per item is: "<<
Invoice2.getItemPrice()<< endl;
cout<<"Invoice2's total amount
is"<<Invoice2.getInvoiceAmount(
)<<endl;
}
```

```
"\nprice per item is: "<<
Invoice2.getItemPrice()<< endl;
cout<<"Invoice2's total amount
is"<<Invoice2.getInvoiceAmount(
)<<endl;
}

Output:

Invoice1's initial part number
is: ed34
and part description is: Screw
Guage
quantity per item is: 2
price per item is: 30
Invoice1's total amount is:60

Invoice2's initial part number
is: e322
and part description is: Screws
quantity per item is: 10
price per item is: 3
Invoice2's total amount is30
```

```
1  // print array using recursion
2
3  #include <iostream>
4  using namespace std;
5  #define MAX_SIZE 100
6  void PrintArray(int arr[], int
     start, int len);
7  int main()
8  {
9  int arr[MAX_SIZE];
10 int num, i;
11  cout<<"size of the array:";
12 cin>>num;
13  cout << " enter  elements in an
     array";
14 for(i=0; i<num; i++)
15 {
16 cin>>arr[i];
17 }
18  cout<<"Elements in the array:
     ";
19 PrintArray(arr, 0, num);
20  return 0;
21 }
22  void PrintArray(int arr[], int
```

```
    start, int len)
23  {
24  if(start >= len)
25  return;
26  cout<<arr[start]<<"\t";
27  PrintArray(arr, start + 1, len);
28  }
29
30  Output:
31  size of the array:4
32   enter  elements in an
    arrayElements in the array:
33   1
34   2
35   3
36   Elements in an array : 1 2 3
37
```

```cpp
// Recursive function to
    reverse a string

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

// Recursive function to
    reverse a given string
// Note string is passed as
    reference parameter
void reverse(string &str, int l
    , int h)
{
    if (l < h)
    {
        swap(str[l], str[h]);
        reverse(str, l + 1, h -
            1);
    }
}

```

```
int main()
{
    string str = "Urvashi Bali"
        ;

    reverse(str, 0, str.length
        () - 1);
    cout << "Reverse of the
        given string is : " <<
        str;

    return 0;
}
Output:
Reverse of the given string is
    :ilaB ihsavrU
```

```cpp
// Recursive function to print minimum element in an array

#include<iostream>
using namespace std;
int getMin(int arr[], int n)

{
  int res = arr[0];

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

        res = min(res, arr[i]);

    return res;
}

 int main()

{
 int arr[] = { 12, 1234, 45, 67, 1 };
 int n = sizeof(arr) / sizeof(arr[0]);
 cout << "Minimum element of
```

```
22    cout << "Minimum element of
      array: " << getMin(arr, n) <<
      "\n";
23        return 0;
24    }
25
26    Output:
27    Minimum element of array: 1
28
29
```

```cpp
// Overload the == and != operators to allow comparisons of complex numbers

#include <iostream>
using namespace std;
class Complex

{

private:

    int real;

    int imag;

public:

    Complex(const int& r, const int& i)
   : real{ r }, imag{ i }
    {}
   friend bool operator == (const Complex &c1, const Complex &c2);
```

```
22      friend bool operator!=
(const Complex &c1, const
Complex &c2);
23
24  };
25  bool operator== (const Complex
&c1, const Complex &c2)
26
27  {
28    return (c1.real == c2.real &&
c1.imag == c2.imag);
29
30  }
31  bool operator!= (const Complex
&c1, const Complex &c2)
32  {
33   return !(c1 == c2);
34  }
35
36  int main()
37
38  {
39      Complex comp1(2,5);
40      Complex comp2(2,3);
41   if (comp1== comp2)
```

```
42      cout<< " Both complex  no
    are same";
43  if (comp1 != comp2)
44      cout<< "Both complex no are
    not same";
45   return 0;
46
47  }
48
49  Output:
50  Both complex no are not same
51
52
```

assignment. Example - when you push codes of this assignment, they should be inside Assignment3 folder.

- Please keep in mind that you don't commit all the codes together. Keep on committing codes module wise or question wise whatever seems available.

1. Modify the Complex class program to enable input and output of complex numbers via overloaded >> and << operators, respectively.
2. Overload the multiplication operator to enable multiplication of two complex numbers as in algebra.
3. Overload the == and != operators to allow comparisons of complex numbers.
4. Write a complete program that prompts the user for the radius of a sphere, and calculates and prints the volume of that sphere. Use an inline function sphereVolume that returns the result of the following expression: (4.0 / 3.0 * 3.14159 * pow(radius, 3)).
5. A parking garage charges a $2.00 minimum fee to park for up to three hours. The garage charges an additional $0.50 per hour for each hour or part thereof in excess of three hours. The maximum charge for any given 24-hour period is $10.00. Assume that no car parks for longer than 24 hours at a time. Write a program that calculates and prints the parking charges for each of three customers who parked their cars in this garage yesterday. You should enter the hours parked for each customer. Your program should print the results in a neat tabular format and should calculate and print the total of yesterday's receipts. The program should use the function calculateCharges to determine the charge for each customer. Your outputs should appear in the following format:

Car	Hours	Charge
1	1.5	2.00
2	4.0	2.50
3	24.0	10.00
Total	29.5	14.50

6. Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four data members—a part number (type string), a part description (type string), a quantity of the item being purchased (type int) and a price per item (type int). Your class should have a constructor that initializes the four data members. Provide a set and a get function for each data member. In addition, provide a member function named getInvoiceAmount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as an int value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0. Write a test program that demonstrates class Invoice's capabilities.
7. Modify the Program-6 to separate the interface and implementation for the purpose of reusability. You should break the program into three different parts as discussed in lectures.
8. A right triangle can have sides that are all integers. A set of three integer values for the sides of a right triangle is called a Pythagorean triple. These three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse. Find all Pythagorean triples for side1, side2 and hypotenuse all no larger than 500. Use a triple-nested for loop that tries all possibilities. This is an example of brute force computing. You'll learn in more advanced computer science courses that there are many interesting problems for which there's no known algorithmic approach other than sheer brute force.

9. A prime integer is any integer that is evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows:
    a. Create an array with all elements initialized to 1 (true). Array elements with prime sub- scripts will remain 1. All other array elements will eventually be set to zero. You'll ignore elements 0 and 1 in this exercise.
    b. Starting with array subscript 2, every time an array element is found whose value is 1, loop through the remainder of the array and set to zero every element whose subscript is a multiple of the subscript for the element with value 1. For array subscript 2, all elements beyond 2 in the array that are multiples of 2 will be set to zero (subscripts 4, 6, 8, 10, etc.); for array subscript 3, all elements beyond 3 in the array that are multiples of 3 will be set to zero (subscripts 6, 9, 12, *15*, etc.); and so on.

    When this process is complete, the array elements that are still set to one indicate that the subscript is a prime number. These subscripts can then be printed. Write a program that uses an array of 1000 elements to determine and print the prime numbers between 2 and 9999. Ignore element 0 of the array.
10. A palindrome is a string that is spelled the same way forward and backward. Examples of palindromes include "radar" and "able was i ere i saw elba." Write a recursive function testPalindrome that returns true if a string is a palindrome, and false otherwise. Note that like an array, the square brackets ([]) operator can be used to iterate through the characters in a string.
11. Write a recursive function printArray that takes an array, a starting sub- script and an ending subscript as arguments, returns nothing and prints the array. The function should stop processing and return when the starting subscript equals the ending subscript.
12. Write a recursive function stringReverse that takes a string and a starting subscript as arguments, prints the string backward and returns nothing. The function should stop processing and return when the end of the string is encountered. Note that like an array the square brackets ([]) operator can be used to iterate through the characters in a string.
13. Write a recursive function recursiveMinimum that takes an integer array, a starting subscript and an ending subscript as arguments, and returns the smallest element of the array. The function should stop processing and return when the starting subscript equals the ending subscript.

Reference - C++ How To Program By Deitel and Deitel