Ajay S

OOP Using C++

|Assignments |

Some Guidelines

- ➔ **Decompose** your task, *recursively*
- ➔ Use **whitespace** Consistently
- ➔ **Use constants**, no magic numbers

  Example: const int MAX_ELEMENT = 20;

  int mystack[MAX_ELEMENT];
- ➔ Use ***nouns*** for **variable** names as they indicate some object
- ➔ Use ***verbs*** for **function** names as they indicate some action
- ➔ Use good, **descriptive names** *(compiler never bites you for longer names, okay!)*

  Example:

  - totalCost is better than variable name **c**
  - CalculateSum() is better than Sum()
- ➔ Make sensible use of **comments.** Describe in short every function, structure and class – the purpose of it, parameters and members etc.
- ➔ **Compilation:**

  To Compile your program with highest C++ standards, use below option.

  g++ -Wall –Werror filename.cpp

  If you see any warning, do fix it!

Program 1

Write a Program to print the message Hello World three times on the screen. Use a for loop.

Program 2

Write a Program to find factorial of a number. Read the input from the keyboard.

Program 3

Write a program to swap two variables using call by reference mechanism.

Program 4

Write a program that reads a number num from keyboard. Write function to print first num number of Fibonacci numbers.

Each Fibonacci number is the sum of the two preceding ones. The sequence starts out 0, 1, 1, 2, 3, 5, 8, ...

Program 5

Write a program that reads a number num from keyboard. Write function to print first num number of prime numbers.

Program 6

Write a program to read a number num from user. Write a function that returns true if the number is a left truncatable prime?

A left truncatable prime number is a number which is prime and remains prime after its leftmost digits are removed one by one. For example 137 is a prime, 37 is a prime and 7 is a prime. So, 137 is a left truncatable prime.

Program 7

Write a program to read a number num from user. Write a function that returns true if the number is a right truncatable prime?

A right truncatable prime number is a number which is prime and remains prime after its rightmost digits are removed one by one. For example, 233 is a prime, 23 is prime, and 2 is a prime. So 233 is right truncatable prime.

Program 8

There are 15 primes which are both left-truncatable and right-truncatable. They have been called two-sided primes. Write a program to find first 10 such prime numbers.

The output of the program should be 2, 3, 5, 7, 23, 37, 53, 73, 313, 317

Program 9

Write a program to print first 8 prime numbers and their factorials using recursive method.

Program 10

Write a function to count the characters in a string. Read the line from the user. Line could contain whitespaces also. Do not use gets().

Program 11

Write a program to implement the atoi() function. Read the string from user. Pass it to a function. The function should convert the string into an int and return.
Example: int x = myAtoI("1232"); // after this call x should be 1232

Program 12

Write a function that accepts a string and returns true if it is a palindrome. Ignore case.
Examples: madam, malayalam are palindromes.
Was it a car or a cat I saw is a palindrome too.

Program 13

Write a program which works as a calculator.
Support Addition, subtraction, multiplication and division, modulus operation.
Write separate user defined function for each operation.

Program 14

Currency Breaker

If an amount is told to the program, the amount should be broken into different smallest possible bills (we call it "note" in India).

Example

amount = 254

Output:

```
1000 x 0 = 0
500 x 0 = 0
100 x 2 = 200
50 x 1 = 50
20 x 0 = 0
10 x 0 = 0
```

```
5 x 0   = 0
1 x 4 = 4
----------------
Total = 254
```

Conditions:

Amount should be read from the command prompt.

Assumptions:

Input entered shall be only in integer format.

Program 15

Friend List v1.0

Write a ***C style*** program that creates and prints the list of friends of a person. Use the following specification.

Example

Name	Friends					
Ajay	Ashish	Vijay	Satish	Ramu	Tom	
Ashish	Santa	Vinod	Anil	Vijay		
Vijay	Anil	Sunil	Ajay	Tom	Amit	Naveen
Rahul	Satish	Sunil	Ajay			

```
struct Person {
            char name[10];
            // add a member here which holds a list of Person variables (you can assume
            max 10 friends for today)
            // dynamic memory allocation shall be highly appreciated instead of max 10.
};
```

Program 16

Write a program to validate the input given on a command line. Provide a number from the command line. The program should validate the input and print the square of the number. If the input contains a character, give an error

Program 17

Below is a Program to find sum of the squares. FIND A BUG IN THE FOLLOWING CODE.

```
#include<iostream>
#include<cmath>
int x , num;
int f(int);
int main()
{
cout << "enter a number: ";
cin >> num;
cout << f(num);
return 0;
}
int f(int s)
{
for(x =1; x <= s; x++) {
int Ans2;
int y;
int Ans[10];
Ans[y] = pow(x,2);
Ans2 = Ans[y] + Ans[y];
cout << Ans2;
y++;
}
return 0;
}
```

Program 18

Character Eraser v1.0

Write a function to remove all occurrences of a character from a string.

Prototype

char* RemoveAllOccurences(char* input, char ch);

Example:

input	Mumbai Bangalore
ch	A
result	Mumbi Bnglore

Program 19

```
int x = 100;
int main()
{
     int x = 200;
     if (x == 200) {
                        cout << ::x << endl;
     }
     return 0;
}
```

Program 20

```
int main()
{
     int x = 200;
     if (x == 200) {
                        cout << ::x << endl;
     }
     return 0;
}
```

Program 21

Write a Program to find the sizes of all the types of data on Linux. Check if the following kind of statements work

```
cout << sizeof(int);
```

Program 22

```
#include<iostream>
using namespace std;
// Program to demonstrate cin and getline()
int main()
{
      char myName[20];
      cin >> myName;
      cout << myName << endl;
      fflush(stdin);

      cin.getline(myName, 20);
      cout << myName << endl;
      return 0;
}
```

Program 23

```
int I = 20;
int main()
{
     int I = 5;
```

```
    cout << I << endl << ::I << end;
    {
                       int I = 10;
                       cout << endl << I << endl << ::I << endl;
    }
    return 0;
}
```

Program 24

```
const int I = 10;
int main()
{
    const int I = 20;

    cout << I << "\t" << ::I << endl;
    cout << &I << " \t" << &::I  << endl;
    return 0;
}
```

Program 25

Write a program to compare two strings. Return true or false based on the comparison. Define your own function to compare the strings.

Prototype:

bool myStringCompare(const char *str1, const char * str2);

Program 26

Write a Program to create class **Animal**. Use scope resolution operator and inline.

Declaration

class Animal

data members		**member functions**
		void set details(char *, int, char *, bool, bool);
char	name[20]	void display();
int	no_of_legs	void change_food_habit();
char	color[10]	
bool	has_tail	
bool	is_veg	

Program 27

Write a program to find larger number of the two numbers. Write the function **get_larger()** in such a way that it accepts both numbers. After comparison the function should return the larger number.

Program 28

Write a program to find the volume of a box when the dimensions are given. Pass the dimensions (length, breadth and height) as arguments to the function. Return the volume from the function to the caller. The name of the function should be **getVolume**.

Program 29

Compile the following program and observe the result carefully.

```
int add_numbers(int, int=0 , int=0);

int main()
{
        int x =10, z = 30;
        int y = 20;
        cout << add_numbers(x,y);

        return 0;
}
int add_numbers(int a, int b, int c)
{
        return (a+b+c);
}
```

Program 30

Write a program to create employee as as objects of a class. Name of the class should be **Employee**. Data of the employee should contain name, emp_id, basic_salary, hra. SEPARATE Functions should be written to store the details, display the details. Write a functions that calculates the salary of an employee

Program 31

Explain what happens when you compile and execute the following code.

```
void fun(int a, int b=1, int c=2)
{
       cout << a << " " << b
             <<  c << endl;
}
int main()
{
       fun(30);
}
```

Program 32

```
void fun(int a = 10, int b, int c)
{
            cout << a << " " << b
                 <<  c << endl;
}
int main()
{
            fun(200, 30);
}
```

Program 33

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

int divide (int a, int b=2)
{
  int r;
  r=a/b;
  return (r);
}

int main ()
{
  cout << divide (12);
  cout << endl;
  cout << divide (20,4);
  return 0;
}
```

Program 34

Write the class Student with two constructors in it.

Program 35

Write the class Employee with two constructors in it.

Program 36

Write a class **MyTime** as specified below.

```
#ifndef TIME_H
#define TIME_H
```

```
class MyTime {
        protected:
                int hours, mins, seconds;
        public:
                MyTime();
                MyTime(int, int, int);
                ~Time();
                void setHours(int);
                void setMinutes(int);
                void setSeconds(int);
                void ShowCurrentTime();
                void AddSeconds();
                void AddMinutes();
                void AddTime(Time &);
};
#endif
```

Program 37

```
int
main()
{
            int i=5;
            int &j = I ;
            int p = 10;

            j = p;
            cout << I << endl << j << endl;
            p = 20;
            cout << I << endl << j << endl;
            return 0;
}
```

Program 38

```
int
main()
{
            int I = 15;
            const int &j = I ;
            cout << I << endl << j << endl;
            I = 9;
            cout << I << endl << j << endl;
            return 0;


}
```

Program 39

```
int main()
{
                int I = 15;
                const int &j = I ;
                cout << I << endl << j << endl;
                j = 9;
                cout << I << endl << j << endl;
                return 0;
}
```

Program 40

Add the following function change_data to the class Student as below

```
void change_data(int num, char * str) // function to change roll_no and name of a Student
{
          roll_no = num;
          strcpy(ptr_name, str);
}
```

and create the objects as below, in main()

```
Student s1(1, "Laxmikanth");
s1.display();
Student s2 = s1;
s2.display();

s2.change_data(2, "Kiran");

s1.display();
s2.display();
```

Program 41

MyArray

Write a class **MyArray** to create a better and dynamic array. We know static arrays do not have boundary check. Implement class MyArray such that if the user tries to insert/access an element beyond boundary, program restricts the user from doing so. What's more? The MyArray allows one to increase the size dynamically.

```
class MyArray {
Private:
                   int *parr;
                   int size;
Public:
                   MyArray(): size(0), parr(NULL)
                   MyArray(int size): size(size)
                   {        parr = new int [size];
                            For(int I = 0; I < size; i++) {
                            Parr[I] = 0;
                   }
```

```
                void insert(int elem);   // should check the boundary
                void TraverseArray();
                void DoubleTheSize();
};
```

Program 42

Friend List v2.0

Write a program that creates and prints the list of friends of a person. Use the following specification. Specification can be modified, not compromising the requirements.

Example

Name	Friends					
Ajay	Ashish	Vijay	Satish	Ramu	Tom	
Ashish	Santa	Vinod	Anil	Vijay		
Vijay	Anil	Sunil	Ajay	Tom	Amit	Naveen
Rahul	Satish	Sunil	Ajay			

```
class Person {
     private:
                    char     name[10];
                    // add a data member here which is able to hold a list of Person
                    //objects? Refer example.

     public:
                    Person();
                    Person(char *, int);
                    Person AddFriend(Person &friend);
                    void display_friendList();
};
```

Program 43

MyStack

Write a class **MyStack** to create stack of integer. Support push, pop and traverse operations.

Program 44

MyQueue

Write a class **MyQueue** to create a queue.

Program 45

Write a program to find average of three numbers passed to a function. Receive the arguments in the function by C++ reference

Program 46

Write the class Employee. And use dynamically allocated memory for name, city of the objects. Add destructor to the class.

Program 47

Find the output of the following program

```
void duplicate (int & a, int & b, int & c)
{
        a*=2;
        b*=2;
        c*=2;
}

int main ()
{
       int x=1, y=3, z=7;
       duplicate (x, y, z);
       cout << "x=" << x << ", y=" << y << ", z=" <<
z;
       return 0;
}
```

Program 48

Try the following program on the compiler. Read and understand the error. Draw a conclusion.

```
void fun(int &arg)
{
     cout <<  ++arg << endl;
}

int main()
{
     fun(10);
}
```

Program 49

With the class student do the following experiments and note the effects

- **a** make a member function protected
- **b** use two access specifiers at the same time for a member
- **c** Write the complete code in following, reorganized manner. Find the effect on

```
            compilation

class Student;

int main()
{
                      -
}

// define the complete class here
class Student {


};
```

Program 50

Write a program to create **TeaMaker** class. It should contain data like number of spoons use default arguments in the set functions.

Program 51

Develop the class Linear_Array with the specifications below. The specification can be modified without compromising the functionality.

```
class Linear_Array {
     private:
                         int array[20];

     public:
                         Linear_Array();
                         void Set_Array_Data();

                         void Traverse_Array();
                         void Insert_at_position(int element, int
                         position);
                         int Find_element(int element);
                         int get_largest();
                         int get_smallest();
};
```

Program 52

Write a program to create a linked list. A member function should be defined to insert a node at the beginning. Use the following specification. The specification can be modified without compromising the functionality.

```
struct Node {
     int data;
     struct Node *pNext;
};

class Linked_List {
     private:
                    int I;
                    int number;
                    Node start;
                    Node *previous;
                    Node *newnode;

     public:
                    void Create_List(Node *);
                    void Insert_Node_in_begining(Node *);
                    void Show_List(Node *);
};
```

Program 53

Write the class TeaMaker with constructors & Default arguments in it.

Program 54

Try the following code and explain the output / error.

```
const int a;
a = 100;
cout << a <<
endl;
```

Program 55

Write a Program to create class Pizza. Data members should be name, price. Member functions should be get_price() which returns the price of a Pizza object. Use dynamic memory for name. Use constructors and destructor in the class.

Program 56

Write a class Bank_Account in such a way that each object of the class stores information about an account holder. Add necessary data and functions along with the constructors and destructor.

Program 57

Try the following examples. Note the compilation and execution results. Also mention the reason for each one.

```
struct ABC {
      static int a;
      int b;
};
int main()
{
      ABC x1;
      ABC :: a = 20;
      x1.b = 12;

      ABC x2;
      x2.a = 33;

      cout << ABC :: a << endl;
      return 0;
}
```

Program 58

```
class Thames {
      private:
                        int num;
                        Thames ob;
};

int main()
{
      cout << "Hello from Thames" <<
      endl;
      cout << sizeof(Thames) << endl;
}
```

Program 59

```
class ABC {
      static int x;
};

int main()
{
      cout << sizeof(ABC) <<
      endl;
      ABC ob;
      cout << sizeof(ob) <<
      endl;
      return 0;
}
```

Program 60

```
class Thames {
        private:
                            int
                            num;
                            Thames *ptr;
};

int main()
{
        cout << "Hello from Thames" << endl;
        cout << sizeof(Thames) << endl;
}
```

Program 61

```
int main()
{
                int i = 10;
                int j = 30;

                const int &ref1 = i;
                ref1 = 100;

                cout << ref1 << endl;
                return 0;
}
```

Program 62

```
class Thames {
         private:
                const int roll = 0;
};


int main()
{
         Thames
ob;
         return 0;
}
```

Program 63

Try to apply const to a non member / global function. Observe the result.

Program 64

```
class Thames {
        private:
                static int roll = 0;
};

int main()
{
        Thames
ob;
        return 0;
}
```

Program 65

```
class Thames {
        public:
                const static int roll = 20;
};

int main()
{
        cout << Thames :: roll <<
endl;
        return 0;
}
```

Program 66

Try the following examples? Note the observations and reasons behind each one.
Also, do at least two experiments with each code and note the experiment?

```
class cube {
  private:
    int len, br, ht;

  public:
    cube(int i = 1, int j = 1, int k = 1)
    {
      len = i;
      br = j;
      ht = k;
    }

    int volume()
    {
      return len * br * ht;
    }
};

int main()
{
  cube c1(2, 3, 4);
  cube c2;

  cout << c1.volume() << endl;
  cout << c2.volume() << endl;

  return 0;
}
```

Program 67

```
void display(int
x)
{
  cout << x << endl;
}

int main()
{
  char c = 'A';
  int i = 10;

  int &ref = i;

  display(c);
  display(i);
  display(ref);

  return 0;
}
```

Program 68

```
void display()
{
  cout << "Hello from empty display" << endl;
}

void display(int
x)
{
  cout << "received an int : " << x << endl;
}
```

```
void display(float x)
{
  cout << "received a float: " << x << endl;
}

void display(float y, char *str)
{
  cout << "received y = " << y;
  cout << " and str = " << str <<
endl;
}

int main()
{
  display();
  display(33);
  display(3.2f);
  display(44, "Ajay");
  return 0;
}
```

Program 69

```
int abs(int i)
{
  return i < 0 ? -i : i;
}

float abs(float f)
{
  return f < 0 ? -f : f;
}


double abs(double f)
{
  return f < 0 ? -f : f;
}


long abs(long l)
{
  return l < 0 ? -l : l;
}

int main()
{
  cout << abs(10) << endl;

  cout << abs(-10.2) << endl;
  cout << abs(-12L) <<
endl;

  return 0;
}
```

Program 70

Write a program to overload member function of a class. First function should accept two integers and second one should accept variable number of arguments.

Program 71

Write a Program to Overload **main()** function. Observe the result.

Program 72

Write a class Complex_Number whose object represents a complex number. Write the operator functions for addition of two complex numbers and Subtraction of two complex numbers.

Program 73

Write a program to overload = operator. Use the class Student.

Program 74

Write a program to overload << operator for the class counter such that it displays *no_of_events* of an object used as below

```
Counter c1(10);
cout << c1 << endl; // output of this statement should be 10
```

Program 75

Write a program to create two classes A and B, with default constructors that announce themselves (has cout statement). Inherit a new class called C from A and create a member object of B in C, but do not create a constructor for C.

Create an object of class C and observe the result.

Program 76

Create a class called vehicle. It should have functions and data which are commonly presently in all various class vehicles.

Create various classes that are vehicles of different types from this class.
Add more functions and instantiate different objects of these classes.
Test if further inheritance is possible. It should be for at least two derived classes. Justify.

Program 77

Try the following code. Detect the error and rectify.

```
class Thames {
      public:
                  int
                  num;
};

int main()
{
      Thames ob;
      Thames *ptr;
      ptr = & ob;
      ptr.num = 100;
}
```

Program 78

```
class Thames {
      protected:
                  int
                  num;
      public:
                  void display() {
                           cout << "Hello from Thames" <<
                           endl;
                  }
                  void set(int val )
                  {
                           num = val;
                  }
};
class myThames : public Thames {
};

int main()
{
      Thames ob1;
      ob1.set(100);
      ob1.display();

      Thames *ptr;
      ptr = & ob1;
```

```
        ptr ->display();

        myThames ob2;
        ob2.set(200);
        ob2.display();
        myThames *myptr;
        myptr = & ob2;
        myptr ->display();
}
```

Program 79

```
class Thames {
      protected:
                  int num;
      public:
                  void display() {
                              cout << "Hello from Thames" << endl;
                  }
                  void set()
                  {
                              num = 100;
                  }
};
class myThames : public Thames {
};

int main()
{
      myThames ob;
      myThames *ptr;
      ptr = & ob;
      ptr ->set();
      ptr ->display();
      return 0;
}
```

Program 80

Write a program to create a bank database with three separate departments. They should be able to access a common database. The department names should be savings, current and funds. Perform day-to-day banking transactions with savings account and display in the common database, the changes made thorough various departments.

Program 81

Write a program to create a family tree of 5 generations and explain the use of the type of inheritance here.

Program 82

Create class Mathematics. Derive Geometry, arithmetic and algebra from it. Add functionalities specific to these branches of mathematics to each to these derived classes. Perform operations and reflect the last updated values in all the classes.

Program 83

Create a class Monkey which has member function move(). Inherit this class into Human. Override the function move(). Use base class pointer to invoke the function move() for different objects. Observe the result.
Then make the function move() virtual. Observe the changes in the output.

Program 84

Define

```
class Base{
public:
    virtual  void  iAm () { c o u t << "b a s e \ n "; }
};
```

Derive two classes from base , and for each define iAm() to write out the name of the class. Create objects of these classes and call iAm() for them. Assign pointers to objects of the derived classes to base* pointers and call iAm() through those pointers.

Program 85

Create a class Point_2D. Add appropriate data members and member functions. Inherit this class in Point_3D. Create objects both the classes and invoke member functions. Observe the result.

Program 86

```
class Thames {
        public:
                virtual void display() {
                }
};
int main()
{
        cout << sizeof(Thames) << endl;
        return 0;
}
```

Program 87

```
class A {
        public:
                void fun()
                {
                        cout << " A:: fun()" <<
endl;
                }
};

class B : public A {
        public:
                void fun()
                {
                        cout << "B :: fun()" <<
endl;
                }
};

int main()
{
        A *ap;
        B ob;
        ap = &ob;
        (B *)ap->fun();
        return 0;
}
```

Program 88

```
class A {
        public:
                void fun()
                {
                        cout << " A:: fun()" <<
endl;
                }
};

class B : public A {
        public:
                void fun()
                {
                        cout << "B :: fun()" <<
endl;
                }
};

int main()
{
        A *ap;
        B ob;
        ap = &ob;
        ap->B::fun();
        return 0;
 }
```

Program 89

Write **a template function** which accepts an array of ten elements and arranges them in ascending order.

Program 90

Write a **template function to f**ind larger number of the two numbers.
Read the numbers from the keyboard. Write the function **get_larger()** in such a way that it accepts both numbers as arguments. After comparison the function **should return the larger number.**

Program 91

Write a template class to create a **Stack**.

Program 92

Write a template class to create a **Queue**.

Program 93

MyArray templatized :-)

Rewrite the class **MyArray** to templatize it.

Program 94

Write a template class named as MyCalculator. Use multiple typenames wisely and support addition, subtraction, multiplication, division, modulus operations well.

Program 95

Rewrite the class Student. Use ***string*** object to store the name of a student.

Program 96

Crate a ***vector*** of Counter objects.

Program 97

Create a ***string*** object. Check if you can use strlen, strcpy functions on a string object.

Program 98

Write a class which contains a ***string*** as a data member. Pass this object to a global function. In the function, create a char pointer to which allocate the memory dynamically. Copy the string data of the object into this dynamically allocated memory.

Program 99

FIND THE COMPILER ERROR AND REASON BEHIND IT

```
#include<iostream>
#include<vector>
#include<string>
#include<stdexcept>
using namespace std;

template<typename T>
class myStack {
        protected:
                vector <T> elems;
        public:
                void push(T const &);
                //void pop();
                T pop();

                T top() const;
                bool empty() const
                {
                        return elems.empty();
                }
                void clear();
                //void browse();
                //void myStack<T> :: browse()
                void browse()
                {
                    //template<typename T>
                    vector<T> :: iterator ptr;
                     ptr = elems.begin();
                      //while(elems.end() != true){
                           cout << *ptr << endl;
                            ptr++;
                            cout << *ptr << endl;
                    //}
            }
};

template<typename T>
void myStack<T> :: clear()
{
        elems.clear();
}


template<typename T>
void myStack<T> :: push(T const &ob)
{
        cout << "TRACE: push()" << endl;
        elems.push_back(ob);
}
/*
template <typename T>
void myStack <T> :: pop()
*/
template <typename T>
T myStack <T> :: pop()
{
        cout << "TRACE: pop() " << endl;
        if(elems.empty()) {
                throw out_of_range("myStack<> :: pop() on empty stack\n");
        }
        T temp;
        temp = elems.back(); // back() returns an element
        elems.pop_back(); // pop_back() does not return anything
        return temp;
}

template<typename T>
T myStack<T> :: top() const
{
        return elems.back();

}
```

Program 100

FriendList v2.0
Write a program that creates and prints the list of friends of a person. Use the following specification. Specification can be modified, not compromising the requirements.

Example

Name	Friends					
Ajay	Ashish	Vijay	Satish	Ramu	Tom	
Ashish	Santa	Vinod	Anil	Vijay		
Vijay	Anil	Sunil	Ajay	Tom	Amit	Naveen
Rahul	Satish	Sunil	Ajay			

```
class Person {
        private:
                        string      name;
                        int         age;
                        vector<Person *> friendList;
        public:
                        Person();
                        Person(string, int);
                        Person AddFriend(Person &friend);
                        void display_friendList();
};
```

Program 101

Write a program to read <string, int> pair from the user in to a ***multimap***. Print the sum of all values corresponding to each key.
Example:
If the **input** pairs are as below

Ajay	15
Vijay	2
Ajay	12
Vijay	2

Anil	12
Vijay	1
Anil	2

Output:

```
Ajay 27
Anil  14
Vijay  5
```

Program 102

Create a vector. Push 1 to 50 into it. Create another vector. Push the square of each number from first vector into second one. Print contents of vectors side by side.
Example:

```
    1    1
    2    4
    3    9 and so on.
```

Program 103

Given the following function declaration, implement the body of min() to find and return the smallest element of vec using an iterator to traverse vec:

```
template <class T>
T GetMin( const vector<T> &vec );
```

Program 104

Write a program to copy the contents of a file into another file. Names of the source and destination files should be provided on command line.

Program 105

Write a program to average real numbers input from a text file called "input.txt". Each line of the text file will contain a single number to be averaged. Collect and sum numbers from the file until there are no more numbers in the file, then compute the average. Write the sum, the number of entries in the file and the average to another text file called "output.txt".

Program 106

Write a program to write a two-dimensional array to a file and read it back from the file. The file should be a binary file.

Program 107

Write a program to read input from the command line. The input is the time taken by a driver to finish a race. The input shall be in the form of **hours:minutes:seconds**
Store the input in ***string*** object. Program should calculate the number of seconds and display it.

Program 108

Write a program to read an input file which contains an integer each on separate lines. The program should compute the number of unique integers and print it.

Program 109

Write a program to read input file which contains a number each on separate lines. The program should sort the integers and write them back to the same file in descending order.

Program 110

Write a program to read input file which contains a number each on separate lines. The program should find the smallest and largest number in the file. Append the result to the input file itself as in the example below.

Example

Smallest = 3.2
Largest = 1122

Program 111

Character Eraser v2.0

Write a function to remove all occurrences of a character from a string. Use the following prototype.

string RemoveAllOccurences(string input, char ch);

Example:

input	Mumbai Bangalore
ch	A
result	Mumbi Bnglore

Program 112

Cannonballs

Suppose that you have somehow been transported back to 1777 and the Revolutionary War. You have been assigned a dangerous reconnaissance mission: evaluate the amount of ammunition available to the British for use with their large cannon which has been shelling the Revolutionary forces. Fortunately for you, the British—being neat and orderly—have stacked the cannonballs into a single pyramid-shaped stack with one cannonball at the top, sitting on top of a square composed of four cannonballs, sitting on top of a square composed of nine cannonballs, and so forth. Unfortunately, however, the Redcoats are also vigilant, and you only have time to count the number of layers in the pyramid before you are able to escape back to your own troops. To make matters worse, computers will not be invented for at least 150 years, but you should not let that detail get in your way. Your mission is to write a recursive function Cannonball that takes as its argument the height of the pyramid and returns the number of cannonballs therein.

Prototype

int getCannonballs(int height);

Program 113

Given a string, create a function ReverseString that returns the string in reverse order. Consider both recursive and iterative techniques for solving this problem. Which one is easier to come up with?

string ReverseString(string str);

Program 114

Write a program to remove neighboring duplicates from the input.

Example

input	5 6 7 7 8 2 1 22 16 16 2 9 8 8 19
output	5 6 7 8 2 1 22 16 2 9 8 19

Program 115

Debug

```
int main()
{
        int I = 10;
        int j = 20;

        cout << (I > j) ? I : j;
        cout << endl;
        return 0;
}
```

Program 116

See the nature of the following code

```
int I = 10;
int * p = &I;
cout << p << endl;              // prints the address of I
char * str = “Thames” ;
cout << str << endl;            // prints string Thames
char * ptr = str;
cout << ptr << endl;          // also prints Thames
```

Write code to print the address of string.

Program 118

Create a file using notepad or vim. Copy the following poem and paste it in the file. Save the file as Poem.txt.

At The Last Watch

Pity, in place of love,
That pettiest of gifts,
Is but a sugar-coating over neglect.
Any passerby can make a gift of it
To a street beggar,
Only to forget the moment the first corner is turned.
I had not hoped for anything more that day.

You left during the last watch of night.
I had hoped you would say goodbye,
Just say 'Adieu' before going away,
What you had said another day,
What I shall never hear again.
In their place, just that one word,
Bound by the thin fabric of a little compassion
Would even that have been too much for you to bear?

When I first awoke from sleep
My heart fluttered with fear
Lest the time had been over.
I rushed out of bed.
The distant church clock chimed half past twelve
I sat waiting near the door of my room
Resting my head against it,
Facing the porch through which you would come out.

Even that tiniest of chances
Was snatched away by fate from hapless me;

```
I fell asleep
Shortly before you left.
Perhaps you cast a sidelong glance
At my reclining body
Like a broken boat left high and dry.
Perhaps you walked away with care
Lest you wake me up.
Awaking with a start I knew at once
That my vigil had been wasted
I realised, what was to go went away in a moment,
What was to stay behind stayed on
For all time.

Silence everywhere
Like that of a birds' nest bereft of birds
On the bough of a songless tree.
With the lifeless light of the waning moon was now blended
The pallor of dawn
Spreading itself over the greyness of my empty life.
I walked towards your bedroom
For no reason.
Outside the door
Burnt a smoky lantern covered with soot,
The porch smelt of the smouldering wick.
Over the abandoned bed the flaps of the rolled-up mosquito-net
Fluttered a little in the breeze.
Seen in the sky outside through the window
Was the morning star,
Witness of all sleepless people
Bereft of hope.

Suddenly I found you had left behind by mistake
Your gold-mounted ivory walking stick.
If there were time, I thought,
You might come back from the station to look for it,
But not because
You had not seen me before going away.
```

Write a program to print
- the number of stanzas in the poem.
- The number of lines in the poem

Also print the below data for each line.
- Number of words
- Number of special characters
- Number of upper case vowels
- Number of upper case consonants

Program 118

Write an encryption program that reads from cin and writes the encoded characters to cout.

You might use simple encryption scheme where key is a number provided as input. The program uses the number in key, and encrypts the input string. Re-encrypting encoded text with the same key should produce the original text. If no key (or a null string) is passed, then no encryption is done.

Program 119

Write a class in such a way that only one instance is created from it. Use this instance to perform the operations.

Program 120

Write your own implantation of auto_ptr.

Program 121

Write a class Pizza. Derive VegPizza and ChickenPizza from it. Write a global function CreatePizza(int opt) that accepts an option and creates a Veg or Chicken Pizza dynamically and returns the address of the same. (you'll need Pizza* as return type for this function). Pass various options and run the program.

Program 122

Write a program which works as a phonebook. The class should be named as MyPhonebook and there should be only one instance of the class in the program. Using this object, one should be able to perform below operations.

a. Add a contact to the phonebook by passing name and number to a function
   One should be able to store more than one numbers per person. If the name exists, the name of the person should not be repeated. Instead add the number to the existing number (Do not replace, this could be another number of the same person)
b. Search a contact by passing name of the person to a function. The function should provide all the numbers of that person.
c. Edit a contact by passing name of the person to a function. The function should support editing of name or one of the numbers under this name.
d. Delete a contact. Pass the name of the person to a function. The function searches for the contact. If present, the functions should confirm from the user for deletion and then delete all the corresponding numbers of the contact.
e. Display the contacts. A function should display the name and all the numbers of this person. The output of this function should be well formatted.

Program 123

Write a class named as **StringToNumberConverter.** This class has a char array as a member (size 10!). Write a default and parameterized constructor. Implement a function toNumber() as a member that returns an int. Below is a sample case.

```
StringToNumberConverter ob1("4567");   // "4567" is
int num = ob1.toNumber();

cout << num ;       // displays 4567
cout << ++num ;  // displays 4568
```

Program 124

Write a function which accepts an array as argument and returns true if each element of the array is unique, false otherwise. Do not use anything global.
Signature of the function must be as below
bool isEachElementUnique(int* pArr, int numOfElements);

Program 125

Write a class named **MyString** whose object is able to hold character string of any given length. Read and follow below given specification carefully.

Rules:

- Use names as specified only
- Use of STL class string is NOT allowed.
- No static array is allowed for any member
- Use new and delete effectively
- Do not create unnecessary variables / functions / logic
- No member function of the class contains any I/O statement. No cin or cout!

Specification:

a. Have all the required member functions for safe usage of the class.
```
MyString ob1;                    // need default constructor
MyString ob2("BLR pune");    // need parameterized constructor
MyString ob3(ob2)              // need copy constructor

ob1 = ob2;                        // need overloaded assignment operator
```

b. Implement **length**() function to return length of the string contained by the object
Examples:
```
cout << ob1.length();         // output should be 0 (printed after creation above)
cout << ob2.length();         // output should be 8 (there is a space)
cout << ob3.length();        // output should be 8 (there is a space)
```

c. Implement **getStr**() function that returns the character string held by the object
Examples
```
cout << ob2.getStr();        // should display BLR pune
```

d. Implement **replace**() as below.
```
char ch = 'X';
ob2.replace(3, ch);
cout << ob2.getStr();          // should display BLX pune
```

e. Implement **replaceAll**() as below
```
MyString ob4("Bangalore");
ob4.replaceAll('a', 'o');         // replace all a with o
```

```
cout << ob4.getStr();          // Bongolore
```

f. Overload **operator +** as a member which behaves as below.

```
MyString s1("Delhi");
MyString s2("Shimla");

MyString s3;
s3 = s1 + s2;                 // needs operator+
cout << s3.getStr();          // should display DelhiShimla
```

Program 126

Write a program to erase common elements from two vectors.

Specification:

- Create two vectors in main()
- Write a separate function for each of the following operations
  - Storing the input numbers into a vector
  - Erasing the common elements from both the vectors
  - Printing the contents of a vector.

Example:

Before Erase:

Vector 1	10	20	2	22	23
Vector 2	33	2	31	10	

After Erase:

Vector 1	20	22	23
Vector 2	33	31	

Rules

- Use pass by reference when passing vector/s to a function
- main() shall not have any logic. It should be used to only declare the two vectors and call other functions as described above.

Assumption: in a given vector, each element is unique.

Program 127

Write a program to erase uncommon elements from two vectors.

Specification:

- Create two vectors in main()
- Write a separate function for each of the following operations
  - Storing the input numbers into a vector
  - Erasing the uncommon elements from both the vectors
  - Printing the contents of a vector.

Example:

Before Erase:

Vector 1	1	2	4	5	6
Vector 2	2	5	6	10	

After Erase:

Vector 1	2	5	6
Vector 2	2	5	6

Rules

- Use pass by reference when passing vector/s to a function
- main() shall not have any logic. It should be used to only declare the two vectors and call other functions as described above.

Assumption: in a given vector, each element is unique.