1. Write a C++ Application to remove all the comments from all given list of .cpp files.

All .cpp files must be given as command line arguments. If a particular file is not present it must throw an exception and continue with next files.

Note: in .cpp files single line comments would begin with // and multi line comments will begin with /\* and end with \*/

1. Assume that the **bank** maintains two types of account for customers the **Savings account & Current account**. The **saving accounts** provide **compound interest** and **withdraw** facility but no check book facility. The **current account** provides check book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level a service change is imposed.

Create a class that stores **customer name, account number and type of account.** For this derive the classes **CUR\_ACCOUNT** & **SAV\_ACCOUNT** to make them specific to this requirement. Include necessary member functions in order to achieve the following tasks:

1. **Accept deposit from a customer and update the balance.**
2. **Display the balance.**
3. **Compute the deposit interest**
4. **Permit withdraws and updates the balance.**
5. **Check for the minimum balance impose penalty, necessary, and update the balance**

include constructors for giving initial values

use new and delete

1. Write a function to square a number using C++ references.
2. Write a function **power(double m, int n)** to raise a number **m** to the power of number **n** and return the result correctly. Use the default value of **2** for n to make the function calculate squares when this argument is omitted.
3. Modify the above program so that using a same function name we could use different data type **(integer, float, double),** but don’t change the **n’s** data type it has to be integer. Use the concept of **function overloading**.
4. Define a class **String** that could work as a user defined string type. Include **constructor** that will enable us to create an uninitialized string.
   * 1. **String s1**; // String with length 0
     2. And also to initialize an object with a string constant at the time of creation like
        + 1. **String S2(“HELLO WORLD”);**

Create objects of **String** class and initialize the string objects as – no argument constructor, one argument constructor, and a function to initialize the data member of string class (accept the string from the user), and a function to display all the string objects.

1. Write a class Circle to implement

Default constructor

One argument constructor

Destructor

Resize(Radius)

Default value parameter constructor

1. Implement the concept of **copy constructor**.
2. Complete the below program

#include<iostream>

using namespace std;

class Vector

{

int \*v;

intsz;

public:

voidVectorsize(int size ) {

sz=size;

v=new int[size]; //Dynamic memory allocation for pointer v

}

void read(void);

voidshowsum(void);

void release(void) { // To free memory

delete []v;

}

};

main()

{

Vector obj;

int count;

cout<<"\nHow many elements are there in Vector?\n";

cin>>count;

obj.Vectorsize(count);

obj.read();

obj.showsum();

obj.release(); //Explicitly freeing memory

}

1. Write a program to create class with constructor and destructor, which keeps track of number of objects created for that class. Also provide the method to display number of objects created.
2. Create a class **FLOAT** that contains one float member. Overload all the four arithmetic operators so that they operate on the object of **FLOAT**.
3. Modify the above program so that the overloaded operators +, –, \*, /, and assignment operator

Choose the return values for these functions so that expressions can be chained together, and for efficiency.

Ex: **object1=object2+object3-object4\*object5/object6**;

1. Complete the below program

#include <cstring>

#include<iostream>

using namespace std;

class Person

{

private:

int age;

char \*name;

public:

int getAge () const

{ return age;}

const char \*getName() const

{ return name; }

Person (char \* name = NULL, int age =0)

{

this->name=malloc( strlen(name)+1);

strcpy(this->name,name);

this->age=age;

}

Person & operator= (const Person & other);

void show()

{

cout<<name<<"::"<<age<<endl;

}

};

Person &Person::operator= (const Person &other)

{

//write code here

}

Person::Person (const Person & other)

{

\*this=other; //OK, use user-defined assignment operator is invoked

}

void main()

{

Person name1("Prashanth",55);

name1.show();

Person name2("Praveen",66);

name2.show();

name2=name2;

name2=name1;

name2.show();

cout<<"testing"<<endl;

}

1. Write a program to create a date class with month, date and year as members. Provide setters and getters. Overload inserter operator(<<) to display the date class objects.
2. Write a program with the following :
   * 1. A function to read two double type numbers from keyboard
     2. A function to calculate the division of these two numbers
     3. A **try** block to **throw** an exception when a wrong type of data is keyed in.
     4. A **try** block to delete and **throw** an exception if the condition “divide by zero” occurs.
     5. Appropriate **catch** block to handle the exceptions thrown.
3. Create a class called money that has 2 attribute rupee and paisa. Add getter and setter properties. Overload + operator with two implementations (1) that adds two money objects. 2)that adds a money object and a double(where money is given in the form of double, for example 50.30). Throw an exception if paisa attribute is >100 or is not a multiple of 4..
4. Write a program that will create a data file containing the list of telephone numbers. Use a class object to store each set of data.
   * 1. File looks like

AAA 56986

BBBBB 42153

CC 78954

. . . . . . . . . .

. . . . . . . . . .

1. Modify the above program to implement the following tasks:
   1. Determine the telephone number of the specified person.
   2. Determine the name of a telephone number is known.
   3. Update a telephone number whenever there is a change.
2. Create two classes, **A** and **B,** with default constructors that announce themselves. 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 results.
3. Create a class called **Spaceship** with a **fly ( )** method. Inherit **Shuttle** from **Spaceship** and add a **land ( )** method. Create a new **Shuttle**, up cast by pointer or reference to a **Spaceship**, and try to call the **land ( )** method. Explain the results.
4. Consider a graphics system that has classes for various figures—rectangles, squares, triangles, circles, and so on. For example, a rectangle might have data members for height, width, and center point, while a square and circle might have only a center point and an edge length or radius, respectively. In a well-designed system, these would be derived from a common class, Figure. You are to implement such a system. The class Figure is the base class. You should add only Rectangle and Triangle classes derived from Figure. Each class has stubs for member functions erase and draw. Each of these member functions outputs a message telling what function has been called and what the class of the calling object is. Since these are just stubs, they do nothing more than output this message. The member function center calls the erase and draw functions to erase and redraw the figure at the center. Since you have only stubs for erase and draw, center will not do any “centering” but will call the member functions erase and draw. Also add an output message in the member function center that announces that center is being called. The member functions should take no arguments. There are three parts to this project: a. Write the class definitions using no virtual functions. Compile and test. b. Make the base class member functions virtual. Compile and test. c. Explain the difference in results. For a real example, you would have to replace the definition of each of these member functions with code to do the actual drawing.
5. A garments sales outlet needs a solution to manage its Diwali discount offers. The solution should tell the customer, amount he/she would save by availing the discount offer. Design a **Sale** class, which keeps track of **Item Quantity** and **Item UnitPrice**, contain functions to set/get these attributes and a **GetBill()** function which return the **bill-amount**. Derive a class **DiwaliDiscountSale** from **Sale** class, which keeps track of discount being offered – say 10% this season, contain functions to set/get discount.

Write a **ComputeSaving** function which takes two **Sale** objects and determine the savings

based on bill-amount of each sale.

Write a program to use these classes. It should accept number of items and **UnitPrice** from user, **Diwali discount** % from user. Compute the **savings** in case customer avails the discount offer.

1. Create a class **F** with a public function **FF**. Create two derived classes **G** and **H** deriving from **F**. Create one more class **J** deriving from **G** and **H**, add a function **JF** to **J**. Inside **JF** call function **FF**. Observe the behavior. After observing the behavior, change the **G** class inheritance with respect to **F** as **virtual inheritance**. Change the **H** class inheritance with respect to **F** as **virtual inheritance**. Now observe the function **JF** behavior in **J**.
2. Create a class **B** with a virtual member function **VF**, **VF** should simply display letter **B**. Derive two classes **C** and **D** from **B**, and override **VF** in their classes and write out the names **C**, **D** respectively. Create objects of **C** and **D** and call **VF** on each. Create a pointer variable of **B\***. Assign **C**,**D** object pointer variables to pointer **B\*** and call **VF** through **B** pointer variable and observe the **C\*** and **D\*** pointer variables behavior.
3. Write a function **template** that can be used to **swap** two integers or characters.
4. Write a function **template** for finding a **minimum** value contained in an array.
5. Complete the below program

#include<iostream>

using namespace std;

class Vector

{

int \*v;

intsz;

public:

Vector(void){ v =NULL; sz=0;}

Vector (int size) {

}

Vector (Vector & v2); //Copy constructor

~Vector () {

delete []v;

}

int&elem (inti) {

if (i>= sz)

cout<< "Error:out of range\n";

return v[i];

}

void show ();

};

Vector::Vector (Vector &tempv) {

}

void Vector::show () {

}

main ()

{

inti;

Vector obj;

Vector v1 (5), v2 (5);

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

v1.elem(i) = i + 1;

// v1 = v2; // Bit wise copy is done

Vector v3 =v1; //Copy constructor is called v3(v2)

//Vector v3 = v2; //Copy constructor is called v3(v2)

cout<< "\nVector v1:";

v1.show ();

//cout<< "\nVector v2:";

//v2.show ();

cout<< "\nVector v3:";

v3.show ();

Vector v4 = v3; //Copy constructor is invoked v4(v3)

cout<<endl;

v4.show ();

}

1. Write a C++ program to perform following sequence of operation and check the output.

Use STL Stack and Queuse

The five items: A, B, C, D, and E are pushed in a stack. One after the other starting from A. The stack is popped four times and each element is inserted in a queue. Then two elements are deleted from the queue and pushed back on the stack. Now one item is popped from the stack. The popped item is.