**1. Suppose you meet with your friend at a school reunion after a long time. You**

**don’t have the number of your friend. Assume your friends’ name is Shakib. You**

**have given your number, and then your friend also gives his number to you. After**

**some time, he said I had another number. Keep that one also. That’s my official**

**number. I use it during my office hours. So then again, on your phone, you are**

**saving his number under the same contact. Now which OOP features resemble the**

**situation. Write a C++ program on it.**

**Ans:**

**#include <iostream>**

**using namespace std;**

**class information**

**{**

**public:**

**string name;**

**protected:**

**string personal\_number;**

**string office\_number;**

**};**

**class output:public information**

**{**

**public:**

**void printstatus()**

**{**

**cout<<"enter your name:";**

**cin>>name;**

**cout<<"enter your personal number:";**

**cin>>personal\_number;**

**cout<<"enter your office number:";**

**cin>>office\_number;**

**cout<<"name:"<<name<<endl<<"personalnumber:"<<personal\_number<<endl<<"office number:"<<office\_number<<endl;**

**}**

**};**

**int main()**

**{**

**output obj;**

**for(int i=0;i<3;i++)**

**{**

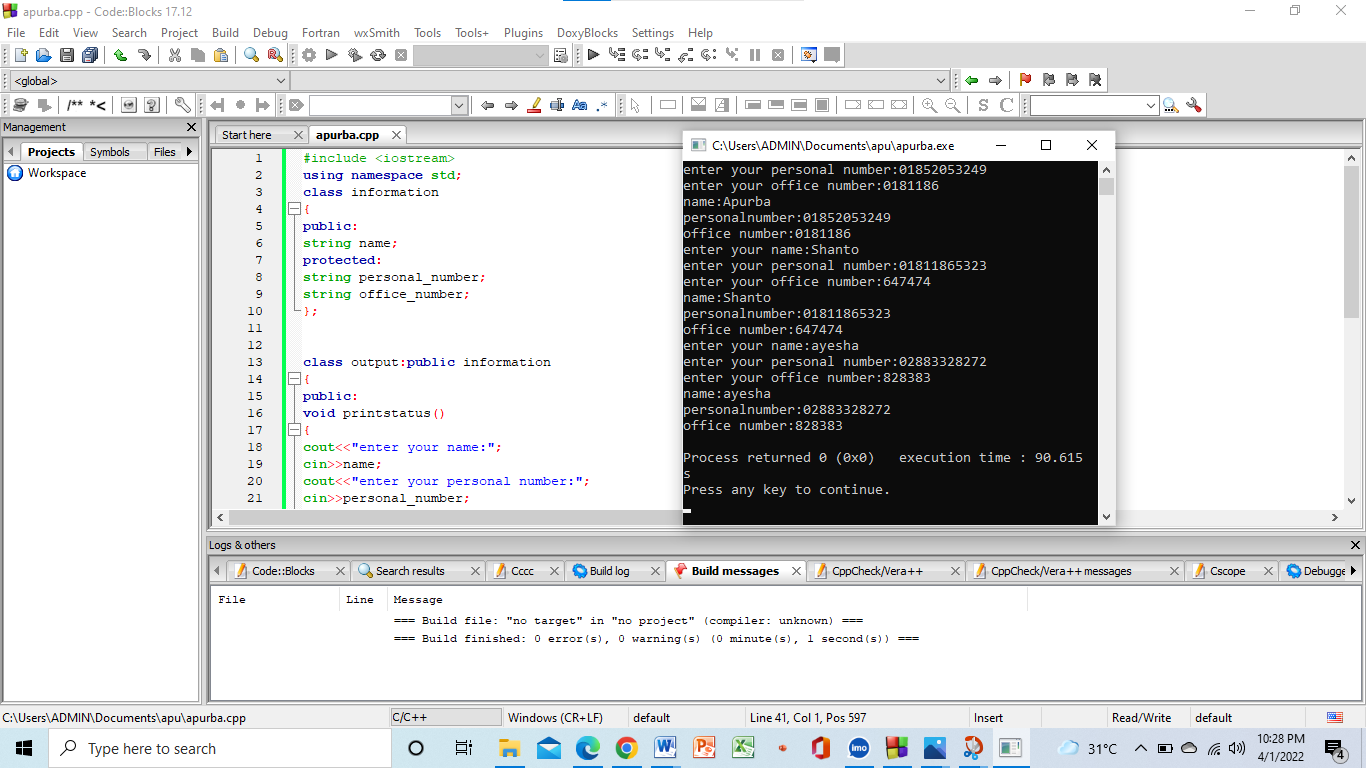
**obj.printstatus();**

**}**

**return 0;**

**}**

**Output:**

****

**2.Write a C++ program that reads a floating-point number. If the number is zero, it**

**prints "zero"; otherwise, print "positive" or "negative.” Add "small" if the absolute**

**value of the number is less than one or "large" if it exceeds 1,000,000.**

**Test Data**

**Input a number: -2534**

**Expected Output:**

**Negative**

**Ans:**

**#include<bits/stdc++.h>**

**using namespace std;**

**int main()**

**{**

**double n;**

**cin>>n;**

**if(n==0)**

**cout<<"Zero";**

**else if(n>0)**

**cout<<"Positive";**

**else**

**cout<<"Negative";**

**n=abs(n);**

**if(n<1)**

**cout<<" Add Small"<<endl;**

**else if(n>1000000)**

**cout<<" Add Large"<<endl;**

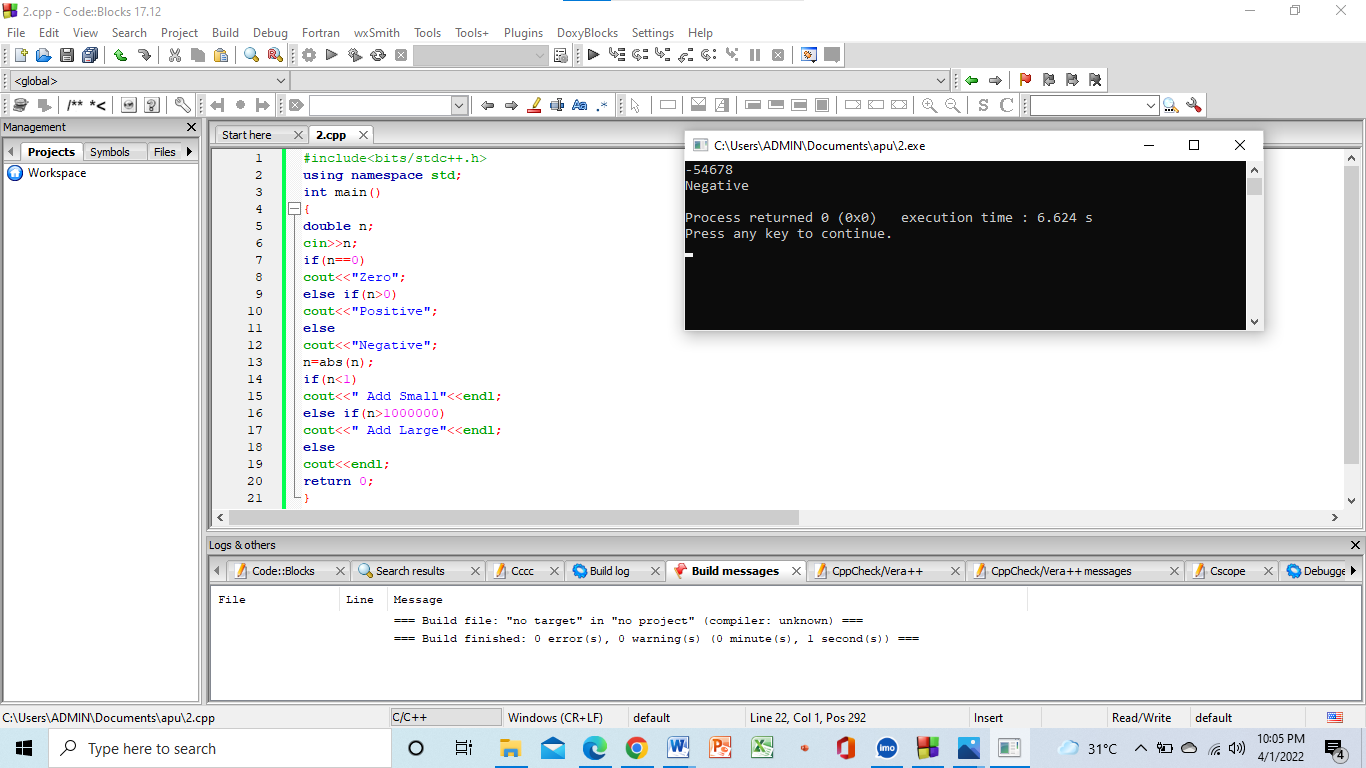
**else**

**cout<<endl;**

**return 0;**

**}**

**Output:**

****

**3.Following is a statement:**

**C++ is an OOP language**

**In this sentence, there are eight vowels and eight constants.**

**Write a C++ program that will take a string as input and count the number of**

**vowels and constants.**

**Ans:**

**#include <iostream>**

**#include<string.h>**

**using namespace std;**

**void stringcount(string);**

**int main()**

**{**

**int i;**

**char s[50];**

**cout<<"Enter any string:";**

**cin>>s;**

**stringcount(s);**

**return 0;**

**}**

**void stringcount(string s)**

**{**

**int i=0,vowel=0,consonants=0;**

**for (i = 0; i < s.length(); i++) {**

**if ((s[i] >= 'a' && s[i] <= 'z') || (s[i] >= 'A' && s[i] <= 'Z')) {**

**if (s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' || s[i] == 'u' || s[i] == 'A' || s[i] == 'E' ||**

**s[i] == 'I' || s[i] == 'O' || s[i] == 'U')**

**{**

**vowel++;**

**}**

**else**

**{**

**consonants++;**

**}**

**}**

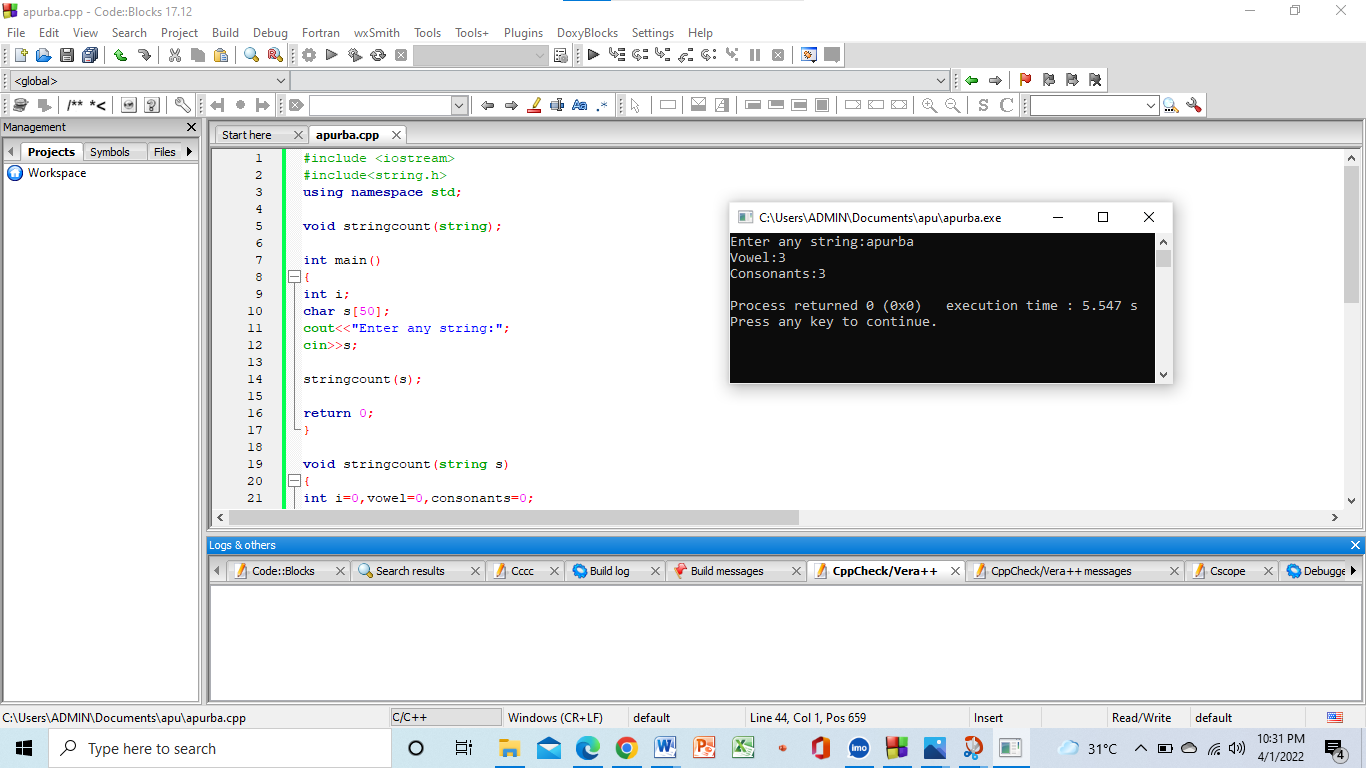
**}**

**cout<<"Vowel:"<<vowel<<endl;**

**cout<<"Consonants:"<<consonants<<endl;**

**}**

**Output:**

****

**4.Suppose you have a savings account in Dhaka bank. In that account, you have**

**some transaction records. Every month you get a deposit in your account, and you**

**also withdraw some money monthly. After all of the month’s transactions, only**

**you get to know the total balance information in your mail sent by the bank**

**authority.**

**Now which OOP features resemble the situation. Write a C++ program on it.**

**Ans:**

**#include<bits/stdc++.h>**

**using namespace std;**

**class Bank{**

**public:**

**string AccountName="0";**

**int Money=0;**

**void create\_account()**

**{**

**cin>>AccountName;**

**cin>>Money;**

**}**

**void deposit(int amount)**

**{**

**Money+=amount;**

**}**

**void withdrawl(int amount)**

**{**

**Money-=amount;**

**}**

**void balance()**

**{**

**cout<<"Account name = "<<AccountName<<endl;**

**cout<<"Total balance = "<<Money<<endl;**

**}**

**};**

**int main()**

**{**

**Bank user;**

**5.In this pandemic situation, the HSC 2020 batch of students got auto passes as there**

**was no other way to take the exam by endangering the lives of those students.**

**Their HSC result was generated from their JSC and SSC. For calculating their**

**GPA, 25 % GPA was taken from JSC, and 75% was taken from SSC.**

**Now write a C++ program where you will have to take input of the marks of a**

**student for the subject English for both JSC and SSC and find out the HSC marks**

**along with the grade of each subject. The Grading Policy is 80-100: Grade A, 70-**

**79: Grade B, 60-69: Grade C, 50-59: Grade D, less than 50 fail.**

**For example:**

**In English, the student got 70 in JSC and 90 in SSC. So, his/her marks in HSC for**

**English will be (70×25%) + (90×75%) =17.5+67.5 =85 in HSC. grade A.**

**Ans:**

**#include<bits/stdc++.h>**

**using namespace std;**

**int main()**

**{**

**double ssc,jsc;**

**cin>>jsc>>ssc;**

**jsc=(25\*jsc)/100;**

**ssc=(75\*ssc)/100;**

**double hsc=jsc+ssc;**

**if(hsc>=80 && hsc<=100)**

**cout<<"A"<<endl;**

**else if(hsc>=70 && hsc<=79)**

**cout<<"B"<<endl;**

**else if(hsc>=60 && hsc<=69)**

**cout<<"C"<<endl;**

**else if(hsc>=50 && hsc<=59)**

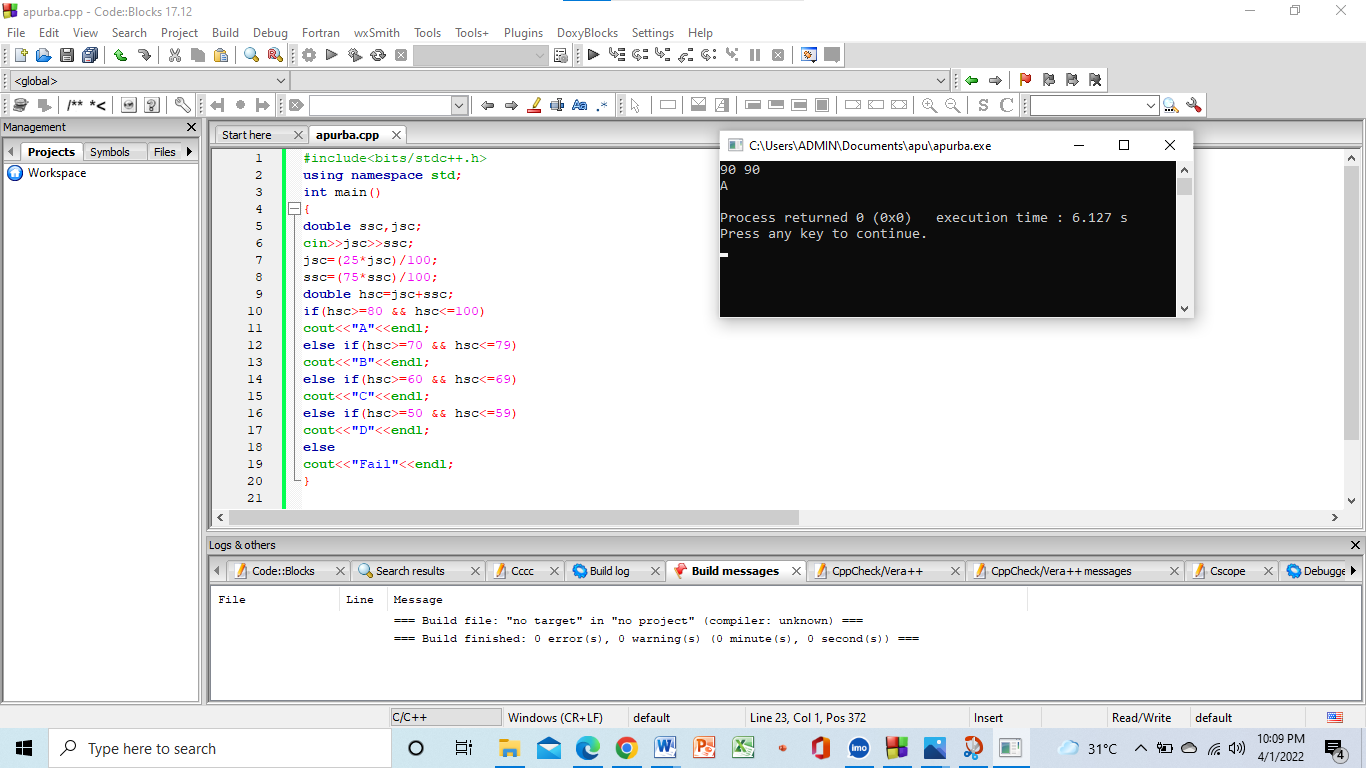
**cout<<"D"<<endl;**

**else**

**cout<<"Fail"<<endl;**

**}**

**Output:**

****

**6.A constructor in C++ is a unique method automatically called when an object of a**

**class is created. A destructor function cleans up any resources allocated to an**

**object. In a class, there are three constructors. To clean up the allocated resources,**

**create a destructor function.**

**Write a C++ program on the above concept where the Name of the Class will be**

**your name. The member variable should show your age and height.**

**Ans:**

**#include<iostream>**

**using namespace std;**

**class Apurba{**

**public:**

**int age;**

**float height;**

**Apurba()**

**{ cout<<"TAKE AGE AND HEIGHT:"<<endl;**

**cin>>age>>height;**

**}**

**Apurba(float h,int a)**

**{**

**float height=h;**

**int age=a;**

**}**

**Apurba(Apurba &obj)**

**{**

**cout<<"AGE:"<<obj.age<<" "<<"Height:"<<obj.height<<endl;**

**}**

**~Apurba()**

**{**

**cout<<"Space Cleaned"<<endl;**

**}**

**};**

**int main()**

**{**

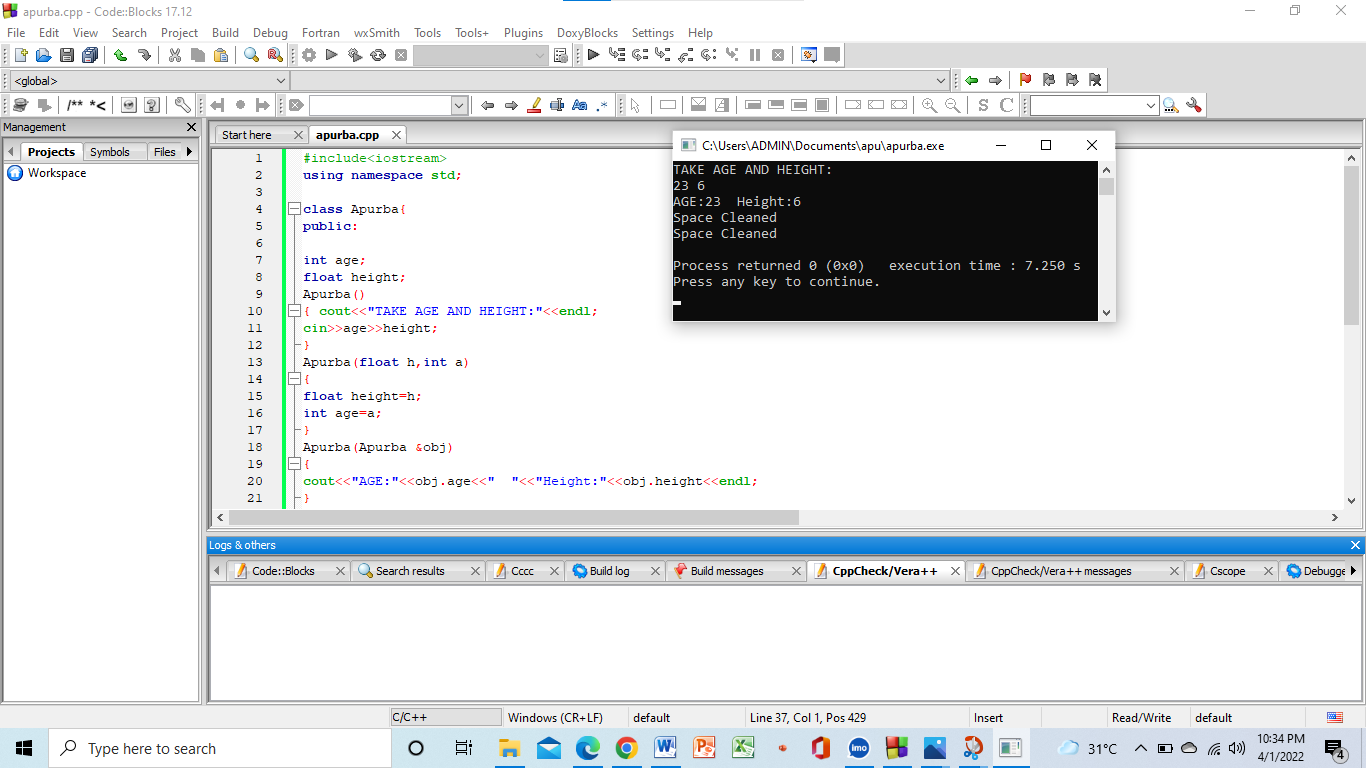
**Apurba obj;**

**Apurba obj1=obj;**

**return 0;**

**}**

**Output:**

****

**7. Write a C++ program to find all pairs of elements in an array whose sum equals a**

**specified number.**

**Ans:**

**#include <iostream>**

**using namespace std;**

**void APURBA(int nums[], int len, int target)**

**{**

**for (int i = 0; i < len - 1; i++)**

**{**

**for (int j = i + 1; j < len; j++)**

**{**

**if (nums[i] + nums[j] == target)**

**{**

**cout<<"Pair found:"<< nums[i]<<","<< nums[j];**

**return;**

**}**

**}**

**}**

**cout<<"Pair not found";**

**}**

**int main()**

**{**

**int nums[7];**

**int target;**

**cout<<"Taking input:"<<endl;**

**for (int i = 0; i <7; ++i) {**

**cin >> nums[i];**

**}**

**cout<<"The target total:"<<endl;**

**cin>>target;**

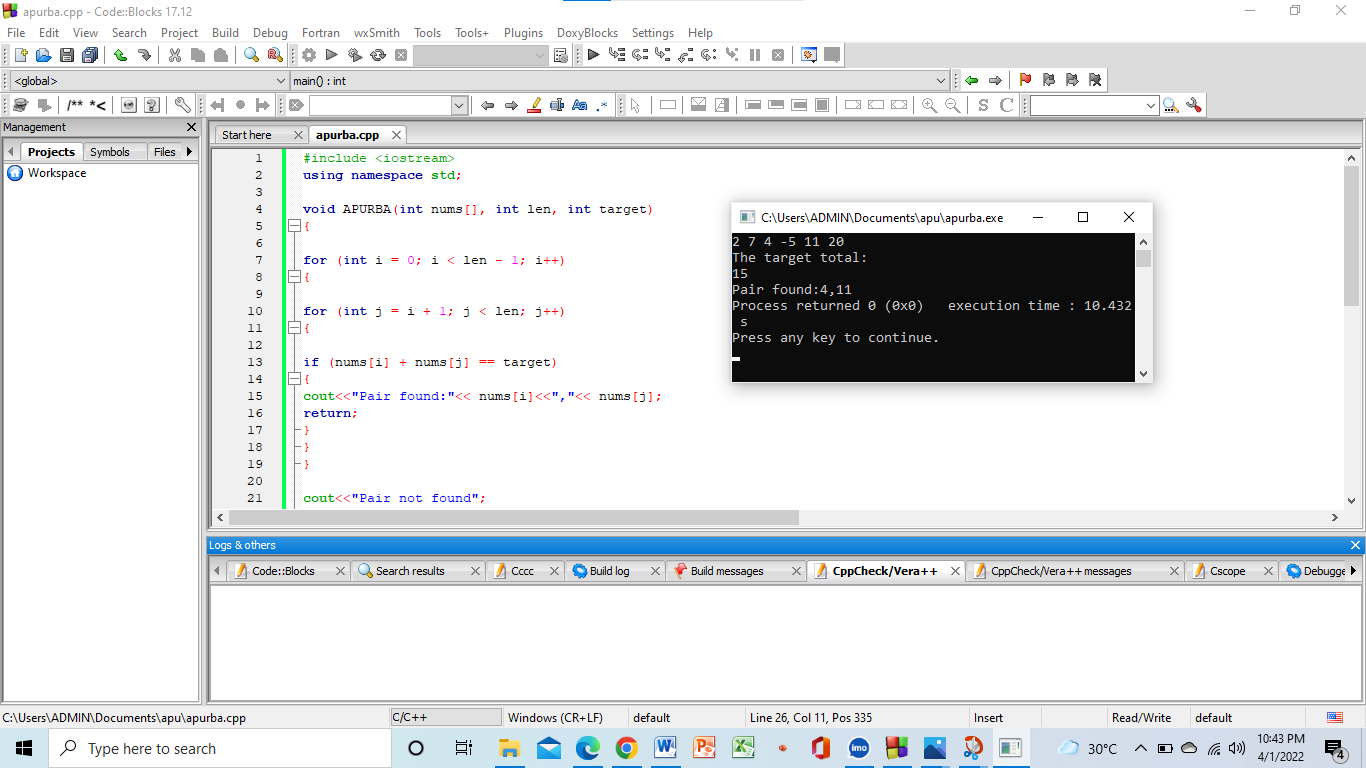
**int len = sizeof(nums);**

**APURBA(nums, len, target);**

**return 0;**

**}**

**Output:**

****

**8. Write a program that accepts three numbers from the user and prints "increasing"**

**if the numbers are in increasing order, "decreasing" if the numbers are in**

**decreasing order, and "Neither increasing nor decreasing order" otherwise. Go to**

**the editor.**

**Test Data**

**Input first number: 1524**

**Input second number: 2345**

**Input third number: 3321**

**Expected Output: Increasing order**

**Ans:**

**#include<bits/stdc++.h>**

**using namespace std;**

**int main()**

**{**

**double a,b,c;**

**cout<<"Input first number:";**

**cin>>a;**

**cout<<"Input second number:";**

**cin>>b;**

**cout<<"Input third number:";**

**cin>>c;**

**if(a<b && b<c)**

**cout<<"increasing"<<endl;**

**else if(c<b && b<a)**

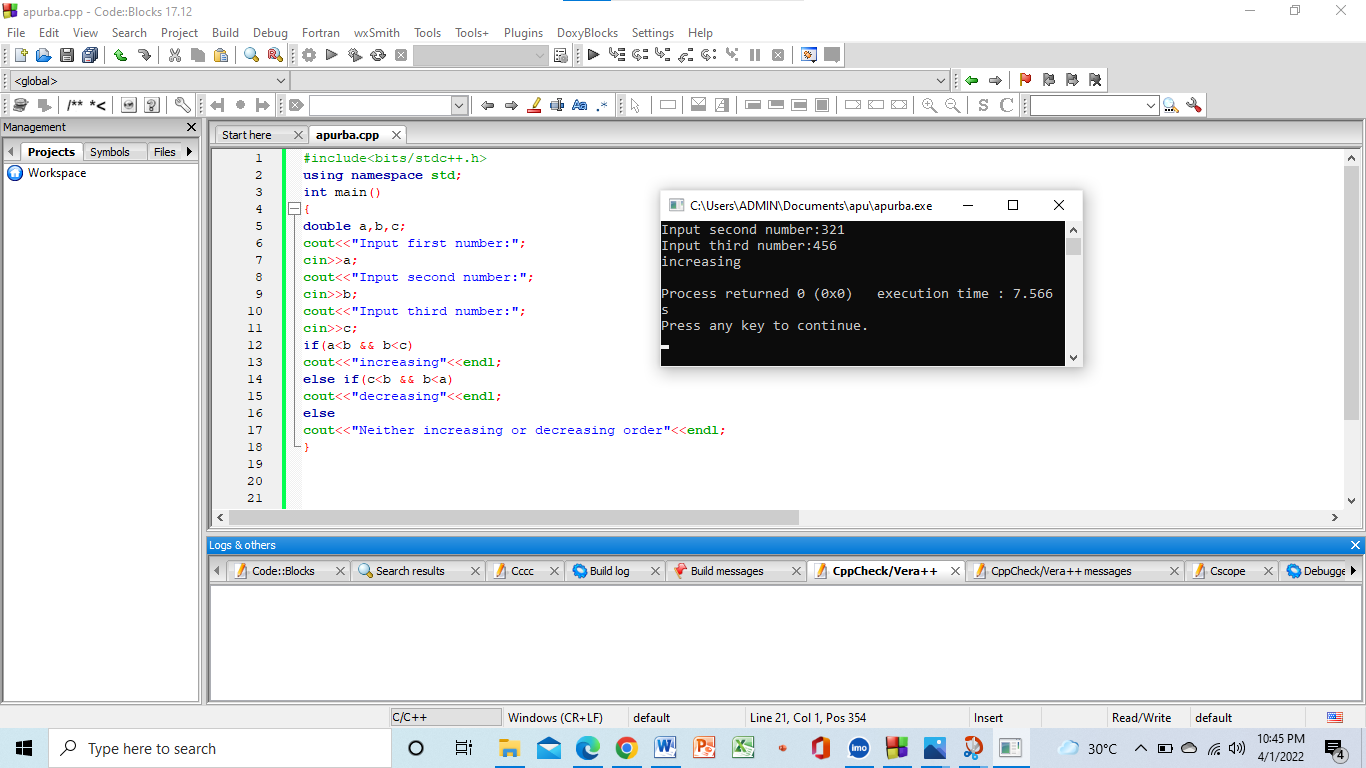
**cout<<"decreasing"<<endl;**

**else**

**cout<<"Neither increasing or decreasing order"<<endl;**

**}**

**Output:**

****

**9.Suppose you go to an Ice Cream Parlor (ABC Ice Cream) near your home one day,**

**buying vanilla-flavored ice cream. A week later, while traveling to the town**

**nearby, you spot another Ice Cream Parlor (of the same chain, ABC Ice Cream).**

**You went to that shop and found a new variant of the Vanilla flavor ice cream,**

**which had a twist of Chocolate flavor. You liked the unique taste. Once back**

**home, you again went to the ice cream parlor near your home to get that fantastic**

**new flavor of ice cream, but unfortunately, you couldn't because that was a**

**specialty of the parlor located in the neighboring town. Now which OOP features**

**resemble the situation. Write a C++ program on it.**

**Ans:**

**#include<iostream>**

**using namespace std;**

**class abcice{**

**public:**

**string icecream="all types of icecream";**

**};**

**class a: public abcice**

**{ public:**

**void dis1(){**

**cout<<icecream;**

**cout<<"near my house"<<endl;**

**cout<<"vanila"<<endl;**

**}**

**};**

**class b: public a{**

**public:**

**void dis2(){**

**cout<<ice01cream;**

**cout<<"nearby town"<<endl;**

**cout<<"vanila+chocolate"<<endl;**

**}**

**};**

**int main()**

**{**

**b obj;**

**abcice obj1;**

**obj1.icecream;**

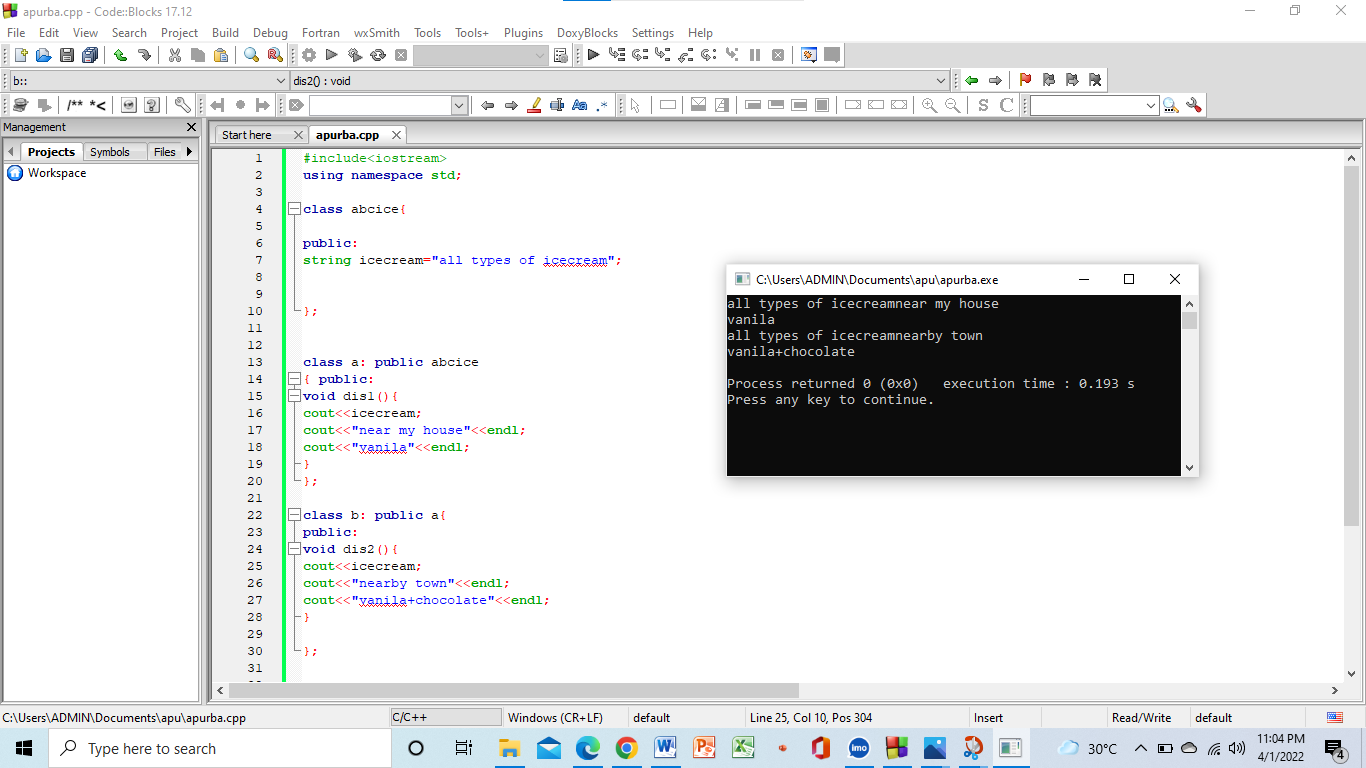
**obj.dis1();**

**obj.dis2();**

**return 0;**

**}**

**Output:**

****

**10.Create a Class called TestClass with a method called Batch.**

**Create another class called ClassOne. And create a method with the name**

**SectionA and inherit the Class TestClass.**

**Create another class called ClassTwo. And make two methods with the name**

**SectionC and SectionD. And inherit the Class TestClass.**

**Now access all the classes from another class named School. Create an object of**

**the three types and access the methods from each category.**

**Ans:**

**#include<iostream>**

**using namespace std;**

**class testclass{**

**public:**

**void batch(){**

**cout<<"we are students"<<endl;**

**}**

**};**

**class classone: public virtual testclass{**

**public:**

**void sectionA(){**

**cout<<"shakil sir"<<endl;**

**}**

**};**

**class classtwo: public virtual testclass{**

**public:**

**void sectionC(){**

**cout<<"saheen sir"<<endl;**

**}**

**void sectionD(){**

**cout<<"Lamia mam"<<endl;**

**}**

**};**

**class school: public classone,public classtwo{**

**public:**

**school(){**

**cout<<"we are one"<<endl;**

**}**

**};**

**int main()**

**{**

**school obj;**

**classone obj1;**

**classtwo obj2;**

**testclass obj3;**

**cout<<"School class output:"<<endl<<endl;**

**obj.sectionA();**

**obj.sectionC();**

**obj.sectionD();**

**cout<<endl;**

**cout<<"ClassOne output:"<<endl;**

**obj1.batch();**

**obj1.sectionA();**

**cout<<endl;**

**cout<<"ClassTwo output:"<<endl;**

**obj2.batch();**

**obj2.sectionC();**

**obj2.sectionD();**

**cout<<endl;**

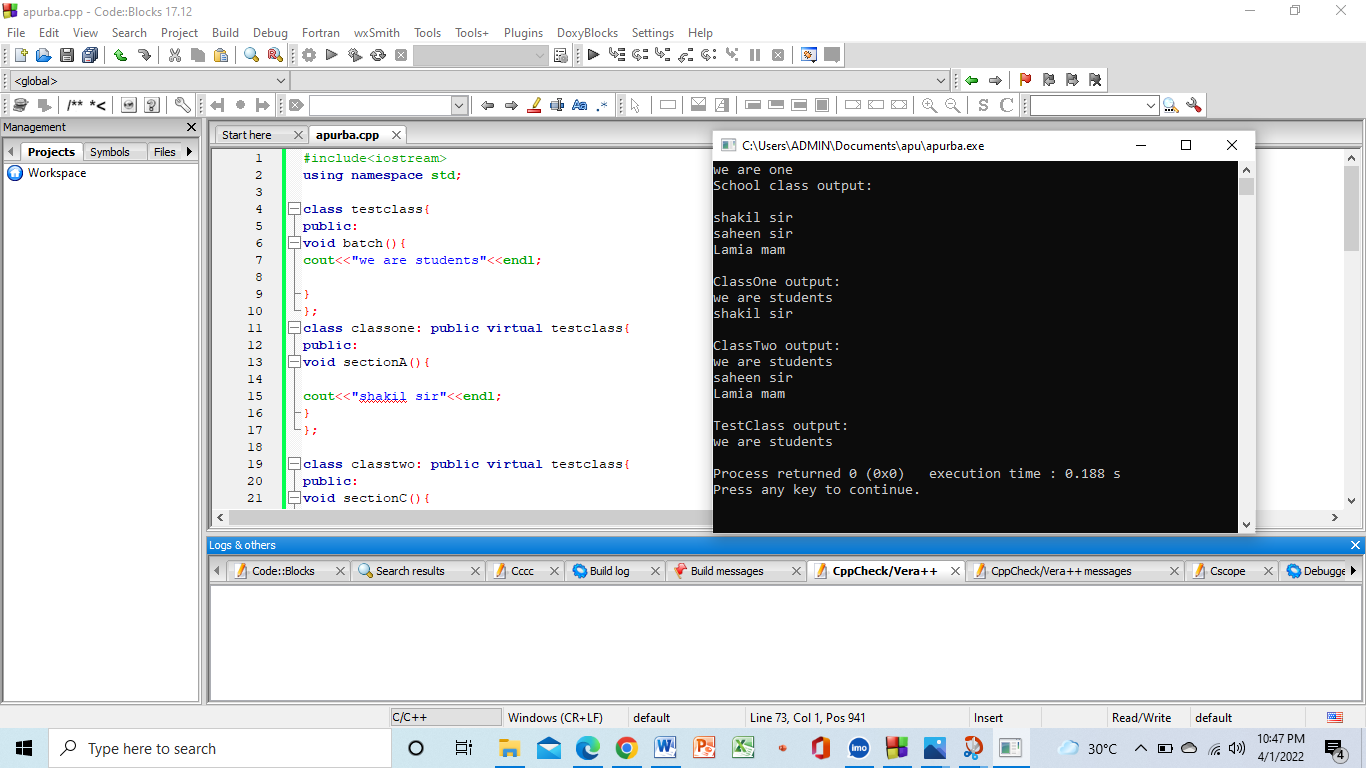
**cout<<"TestClass output:"<<endl;**

**obj3.batch();**

**return 0;**

**}**

**Output**

**:**

**11. We want to store the information about different vehicles. Create a class named**

**Vehicle with two data members called mileage and price. Create its two subclasses**

**\*Car with data members to store ownership cost, warranty (by years), seating**

**capacity, and fuel type (diesel or petrol).**

**\*Bike with data members to store the number of cylinders, number of gears, cooling**

**type (air, liquid, or oil), wheel type (alloys or spokes), and fuel tank size (in inches)**

**Make another two subclasses, Audi and Ford of Car, each having a data member to**

**store the model type. Next, make two subclasses, Bajaj and TVS, each having a data**

**member to store the make-type.**

**Now, store and print the information of an Audi and a Ford car (i.e., model type,**

**ownership cost, warranty, seating capacity, fuel type, mileage, and price.) Do the**

**same for a Bajaj and a TVS bike.**

**Ans:**

**#include<iostream>**

**using namespace std;**

**class vehicle**

**{**

**public:**

**int mileage;**

**int price;**

**};**

**class car:public vehicle**

**{**

**public:**

**int ownership\_cost;**

**int warrenty;**

**int seating\_capacity;**

**string fuel\_type;**

**};**

**class bike:public vehicle**

**{**

**public:**

**int cylinders;**

**int gears;**

**string cooling\_type;**

**string wheel\_type;**

**float fuel\_tank\_size;**

**};**

**class Audi:public car**

**{**

**public:**

**string model;**

**Audi()**

**{**

**cin>>ownership\_cost;**

**cin>>warrenty;**

**cin>>seating\_capacity;**

**cin>>fuel\_type;**

**cin>>mileage;**

**cin>>price;**

**cin>>model;**

**}**

**void print1()**

**{**

**cout<<ownership\_cost<<endl;**

**cout<<warrenty<<endl;**

**cout<<seating\_capacity<<endl;**

**cout<<fuel\_type<<endl;**

**cout<<mileage<<endl;**

**cout<<price<<endl;**

**cout<<model<<endl<<endl;**

**}**

**};**

**class Ford:public car**

**{**

**public:**

**string model;**

**Ford()**

**{**

**cin>>ownership\_cost;**

**cin>>warrenty;**

**cin>>seating\_capacity;**

**cin>>fuel\_type;**

**cin>>mileage;**

**cin>>price;**

**cin>>model;**

**}**

**void print2()**

**{**

**cout<<ownership\_cost<<endl;**

**cout<<warrenty<<endl;**

**cout<<seating\_capacity<<endl;**

**cout<<fuel\_type<<endl;**

**cout<<mileage<<endl;**

**cout<<price<<endl;**

**cout<<model<<endl<<endl;**

**}**

**};**

**class Bajaj:public bike**

**{**

**public:**

**string model;**

**Bajaj()**

**{**

**cin>>cylinders;**

**cin>>gears;**

**cin>>cooling\_type;**

**cin>>wheel\_type;**

**cin>>fuel\_tank\_size;**

**cin>>mileage;**

**cin>>price;**

**cin>>model;**

**}**

**void print3()**

**{**

**cout<<cylinders<<endl;**

**cout<<gears<<endl;**

**cout<<cooling\_type<<endl;**

**cout<<wheel\_type<<endl;**

**cout<<fuel\_tank\_size<<endl;**

**cout<<mileage<<endl;**

**cout<<price<<endl;**

**cout<<model<<endl;**

**}**

**};**

**class TVS:public bike**

**{**

**public:**

**string model;**

**TVS()**

**{**

**cin>>cylinders;**

**cin>>gears;**

**cin>>cooling\_type;**

**cin>>wheel\_type;**

**cin>>fuel\_tank\_size;**

**cin>>mileage;**

**cin>>price;**

**cin>>model;**

**}**

**void print4()**

**{**

**cout<<cylinders<<endl;**

**cout<<gears<<endl;**

**cout<<cooling\_type<<endl;**

**cout<<wheel\_type<<endl;**

**cout<<fuel\_tank\_size<<endl;**

**cout<<mileage<<endl;**

**cout<<price<<endl;**

**cout<<model<<endl;**

**}**

**};**

**int main()**

**{**

**cout<<"Enter the info for Audi Car:"<<endl;**

**Audi obj1;**

**cout<<"Enter the info for Ford Car:"<<endl;**

**Ford obj2;**

**cout<<"Enter the info for Bajaj bike:"<<endl;**

**Bajaj obj3;**

**cout<<"Enter the info for TVS bike:"<<endl;**

**TVS obj4;**

**cout<<"All info for Audi Car:"<<endl;**

**obj1.print1();**

**cout<<"All info for Ford Car:"<<endl;**

**obj2.print2();**

**cout<<"All info for Bajaj Bike:"<<endl;**

**obj3.print3();**

**cout<<"All info for TVS Bike:"<<endl;**

**obj4.print4();**

**return 0;**

**}**

**Output:**

