**Q1.** (The EvenNumber class) Define the EvenNumber class for representing an even

number. The class contains:

* A data field value of the int type that represents the integer value stored in the

Object.

* A no-arg constructor that creates an EvenNumber object for the value 0.
* A constructor that constructs an EvenNumber object with the specified value.
* A function named getValue() to return an int value for this object.
* A function named getNext() to return an EvenNumber object that represents the next even number after the current even number in this object.
* A function named getPrevious() to return an EvenNumber object that represents

the previous even number before the current even number in this object Implement the class. Write a test program that creates an EvenNumber object for value 16 and invokes the getNext() and getPrevious() functions to obtain and displays these numbers.

**Source Code:**

#include<iostream>

using namespace std;

class Even\_number{

int x;

public:

Even\_number() { x=0; }

Even\_number(int a) {x=a; }

int getValue();

int getNext();

int getPrevious(); };

int Even\_number::getValue() {

return x; }

int Even\_number::getNext() {

return x+2; }

int Even\_number::getPrevious() {

return x-2; }

int main() {

Even\_number en(16);

int current,next,previous;

current=en.getValue();

next=en.getNext();

previous=en.getPrevious();

cout<< "The next number is:"<<next<<endl<<"The previous number is:"<<previous<<endl;

}

**Output:**



**Q2.** (The Stock class) Design a class named Stock that contains the following:

* A string data field named symbol for the stock's symbol.
* A string data field named name for the stock's name.
* A double data field named previousClosingPrice that stores the stock price for the previous day
* A double data field named currentPrico that stores the stock price for the current time.
* A constructor that creates a stock with specified symbol and name.
* The constant accessor functions for all data field.
* The mutator functions for previousClosingPrice and currentPrice.
* A constant function named getChangePercent() that returns the percentagechanged from previousClosingPrice to currentPrice. Implement the class.

Write a test program that creates a Stock object with the stock symbol MSFT, the name Microsoft Corporation and the previous closing price of 27.5. Set a new current price to 27.6 and display the price-change percentage.

**Source Code:**

#include<iostream>

#include<cstring>

#include<cmath>

using namespace std;

class Stock {

public:

string symbol,name;

double previousClossingPrice,current\_Price;

Stock(string a,string b) {

symbol=a;

name=b; }

void getChangePercent() {

double change;

change=current\_Price-previousClossingPrice;

int parcentag=(change\*100)/previousClossingPrice;

cout << "The parcentage change is:"<<parcentag<<" %"<<endl; }

void getSymbol() {

cout << "The symbol is:"<< symbol<<endl; }

void getName() {

cout << "The name is:"<<name<<endl; }

double previousClosingPrice() {

return previousClossingPrice; }

double currentPrice() {

return current\_Price; } };

int main() {

Stock st("MSFT","Microsoft Corporation");

st.current\_Price=27.6;

st.previousClossingPrice=27.5;

st.getName();

st.getSymbol();

double previous=st.previousClosingPrice();

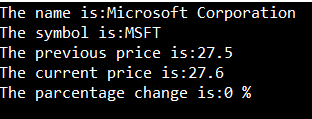
cout << "The previous price is:"<<previous<<endl;

double current=st.currentPrice();

cout << "The current price is:"<<current<<endl;

st.getChangePercent(); }

**Output:**



**Q3.** (The MyDate class) Design a class named MyDate. The class contains:

* The data fields year, month, and day that represent a date. month is 0-based i.e., 0 is for January.
* A no-arg constructor that creates a MyDate object for the current date.
* A constructor that constructs a MyDate object with a specified elapsed time since midnight, January 1, 1970, in seconds.
* A constructor that constructs a MyDate object with the specified year, month, and day.
* Three constant get functions for the data fields year, month, and day, respectively.
* Three set functions for the data fields year, month, and day, respectively.
* A function named setDate(long elapsedTime) that sets a new date for the object using the elapsed time. implement the class. Write a test program that creates two MyDate

objects (using MyDate() andMyDate(3435555513)) and displays their year, month, and

day (Hint: The first two constructors will extract the year, month, and day from the

elapsed time. For example, if the elapsed time is 561555550 seconds, the year is 1987, the month is 9, and the day is 18.)

**Source Code:**

#include<iostream>

using namespace std;

class MyDate {

public:

int year,month,day;

MyDate() {

year=2019;

month=5;

day=13; }

MyDate(long long a) {

year=1970+((((a/60)/60)/24)/365.25);

month=((((a/60)/60)/24)/365.25);

int month1=month;

month=(((((a/60)/60)/24)-(month\*365.25))/30);

float n=((((a/60)/60)/24)-(month1\*365.25));

switch(month) {

case 1:

day=n-31+1;

break;

case 2:

day=n-59+1;

break;

case 3:

day=n-90+1;

break;

case 4:

day=n-120+1;

break;

case 5:

day=n=151+1;

break;

case 6:

day=n-181+1;

break;

case 7:

day=n-212+1;

break;

case 8:

day=n-243+1;

break;

case 9:

day=n-273+1;

break;

case 10:

day=n-304+1;

break;

case 11:

day=n-334+1;

break;

case 0 :

day=n+1; } }

MyDate(int y,int m,int d) {

year=y;

month=m;

day=d; }

void setYear() {

cout << "Enter year:";

cin>>year; }

void setMonth() {

cout << "Enter Month:";

cin >> month; }

void setDay() {

cout << "Enter Day:";

cin >> day; }

int getYear() {

return year; }

int getMonth() {

return month; }

int getDay() {

return day; }

void setDat(long long a) {

year=1970+((((a/60)/60)/24)/365.25);

month=((((a/60)/60)/24)/365.25);

int month1=month;

month=((((a/60)/60)/24)-(month\*365.25)/30);

int n=(((a/60)/60)/24)-(month1\*365.25);

switch(month) {

case 1:

day=n-31;

break;

case 2:

day=n-59;

break;

case 3:

day=n-90;

break;

case 4:

day=n-120;

break;

case 5:

day=n=151;

break;

case 6:

day=n-181;

break;

case 7:

day=n-212;

break;

case 8:

day=n-243;

break;

case 9:

day=n-273;

break;

case 10:

day=n-304;

break;

case 11:

day=n-334;

break;

case 0 :

day=n; } } };

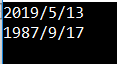
int main() {

MyDate D1,D2(561555550);

cout << D1.getYear()<<"/"<<D1.getMonth()<< "/"<<D1.getDay()<<endl;

cout << D2.getYear()<<"/"<<D2.getMonth()<< "/"<<D2.getDay()<<endl; }

**Output:**



**Q4.** (The MyPoint class) Design a class named MyPoint to represent a point with xand

y-coordinates. The class contains:

* Two data fields x and y that represent the coordinates.
* A no-arg constructor that creates a point (0, 0).
* A constructor that constructs a point with specified coordinates.
* Two get functions for data fields x and y, respectively.
* A function named distance that returns the distance from this point to another

point of the MyPoint type. Implement the class.

Write a test program that creates two points (0, 0) and (10, 30.5) and displays the distance between them.

**Source Code:**

#include<iostream>

using namespace std;

class MyPoint {

public:

double x,y;

MyPoint() { x=0.0; y=0.0; }

MyPoint(double a,double b) { x=a; y=b; }

void getx() {

cout << "The X coordinate is:"<<x<<endl; }

void gety() {

cout << "The Y coordinate is:"<<y<<endl; }

void operator-(MyPoint ob) {

cout << "Distance between X coordinate is:"<< x-ob.x<<endl;

cout << "Distance between Y coordinate is:"<< y-ob.y<<endl; } };

int main() {

MyPoint p1,p2(10.0,30.5);

p1.getx();

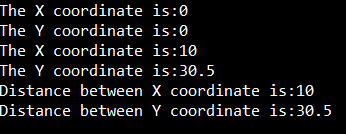
p1.gety();

p2.getx();

p2.gety();

p2-p1; }

**Output:**



**Q5.** Create an Account class to model a bank account. An account has the properties account number, balance, and annual profit rate, date created, and functions to deposit

and withdraw. Create two derived classes for checking and saving accounts.

A checking account has an overdraft limit, but a savings account cannot be overdrawn.

Define a constant virtual toString() function in the Account class and override it in the derived classes to return the account number and balance as a string. Write a test program that creates objects of Account, SavingsAccount, and CheckingAccount

And invokes their toString() functions.

**Source Code:**