**Assignment #03**

**Name: Basharat Hussain**

**Roll No: p17-6102**

**Section: A**

**Date: 29/11/18.**

**Chapter #11**

**Question 01**:

#include<iostream>

#include<string>

using namespace std;

struct movietype{

string m\_name;

string m\_director;

string producer;

int y\_r;

int c\_s;

};

void getinfo();

void displayinfo();

movietype m;

int main()

{

getinfo();

displayinfo();

return 0;

}

void getinfo()

{

cout << "Enter the name of Movie :";

getline(cin, m.m\_name);

cout << "Enter the name of Movie Director :";

getline(cin, m.m\_director);

cout << "Enter the name of Movie Producer :";

getline(cin, m.producer);

cout << "Enter the year movie was released :";

cin >> m.y\_r;

cout << "Enter the number of copies in stock :";

cin >> m.c\_s;

}

void displayinfo()

{

cout << endl << "Name of the Movie :" << m.m\_name << endl;

cout << "Name of the Movie Director :" << m.m\_director << endl;

cout << "Name of Movie Producer :" << m.producer << endl;

cout << "Year movie was released :" << m.y\_r << endl;

cout << "Number of copies in stock :" << m.c\_s << endl;

}

**Question #02**

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

const int NO\_OR\_STUDENTS = 20;

struct studentType

{

string studentFName;

string studentLName;

int testScore;

char grade;

};

void getData(ifstream& inFile, studentType sList[], int listSize);

void calculateGrade(studentType sList[], int listSize);

int highestScore(const studentType sList[], int listSize);

void printResult(ofstream& outFile, const studentType sList[], int listSize);

int main()

{

ifstream inData;

ofstream outData;

studentType studentList[NO\_OR\_STUDENTS];

inData.open("Ch9\_Ex2Data.txt");

if (!inData)

{

cout << "The input file does not exist. Program terminates!"

<< endl;

system("PAUSE");

return 1;

}

outData.open("Ch9\_Ex2Out.txt");

if (!outData)

{

cout << "Cannot open the output file. Program terminates!"

<< endl;

system("PAUSE");

return 1;

}

getData(inData, studentList, NO\_OR\_STUDENTS);

calculateGrade(studentList, NO\_OR\_STUDENTS);

printResult(outData, studentList, NO\_OR\_STUDENTS);

system("PAUSE");

return 0;

}

void getData(ifstream& inFile, studentType sList[], int listSize)

{

for (int i = 0; i < listSize; i++)

inFile >> sList[i].studentFName >> sList[i].studentLName

>> sList[i].testScore;

}

void calculateGrade(studentType sList[], int listSize)

{

int score;

for (int i = 0; i < listSize; i++)

if (score >= 90)

sList[i].grade = 'A';

else if (score >= 80)

sList[i].grade = 'B';

else if (score >= 70)

sList[i].grade = 'C';

else if (score >= 60)

sList[i].grade = 'D';

else

sList[i].grade = 'F';

}

int highestScore(const studentType sList[], int listSize)

{

int score[100];

int highscore = score[0];

for (int i = 0; i < listSize; i++)

{

if (score[i] > highscore)

{

highscore = score[i];

}

}

void printResult(ofstream& outFile, const studentType sList[], int listSize)

{

int maxScore = highestScore(sList, listSize);

int i;

outFile << setw(15) << "Student Name "

<< setw(10) << "Test Score"

<< setw(7) << "Grade" << endl;

for (i = 1; i < listSize; i++)

outFile << left << setw(25)

<< sList[i].studentLName + ", " + sList[i].studentFName

<< right << " " << setw(5) << sList[i].testScore

<< setw(6) << " " << sList[i].grade << endl;

outFile << endl << "Highest Test Score: " << maxScore << endl;

outFile << "Students having the highest test score:" << endl;

for (i = 1; i < listSize; i++)

if (sList[i].testScore == maxScore)

outFile << sList[i].studentLName + ", " + sList[i].studentFName << endl;

}

Question#04

#include<iostream>

#include<string>

using namespace std;

struct menuitem

{

string menulist;

double price;

};

menuitem menu[8];

void getdata();

void showdata();

void selectItems();

void cal();

int c[8] = { 0, 0, 0, 0, 0, 0, 0, 0 };

int main()

{

double t;

getdata();

showdata();

selectItems();

cal();

return 0;

}

void getdata()

{

menu[1].menulist = "Plain Egg";

menu[1].price = 15.00;

menu[2].menulist = "Omelet";

menu[2].price = 15.00;

menu[3].menulist = "Paratha";

menu[3].price = 12.00;

menu[4].menulist = "French Toast";

menu[4].price = 20.99;

menu[5].menulist = "Fruit Basket";

menu[5].price = 120.49;

menu[6].menulist = "Coffee";

menu[6].price = 50.00;

menu[7].menulist = "Tea";

menu[7].price = 20.00;

}

void showdata()

{

cout << "Breakfast items offered by the restaurant are" << endl;

cout << 1 << "\t" << menu[1].menulist << setw(12) << "RS " << menu[1].price << endl;

cout << 2 << "\t" << menu[2].menulist << setw(15) << "RS " << menu[2].price << endl;

cout << 3 << "\t" << menu[3].menulist << setw(14) << "RS " << menu[3].price << endl;

cout << 4 << "\t" << menu[4].menulist << setw(9) << "RS " << menu[4].price << endl;

cout << 5 << "\t" << menu[5].menulist << setw(9) << "RS " << menu[5].price << endl;

cout << 6 << "\t" << menu[6].menulist << setw(15) << "RS " << menu[6].price << endl;

cout << 7 << "\t" << menu[7].menulist << setw(18) << "RS " << menu[7].price << endl;

}

void selectItems()

{

int ch,quantity;

char con;

do{

cout << "Enter your choice :";

cin >> ch;

cout << "Enter the Quantity :";

cin >> quantity;

switch (ch)

{

case 1:

{

c[1] = c[1] + quantity;

cout << "You have Selected :" << menu[1].menulist << endl;

break;

}

case 2:

{

c[2] = c[2] + quantity;

cout << "You have Selected :" << menu[2].menulist << endl;

break;

}

case 3:

{

c[3] = c[3] + quantity;

cout << "You have Selected :" << menu[3].menulist << endl;

break;

}

case 4:

{

c[4] = c[4] + quantity;

cout << "You have Selected :" << menu[4].menulist << endl;

break;

}

case 5:

{

c[5] = c[5] + quantity;

cout << "You have Selected :" << menu[5].menulist << endl;

break;

}

case 6:

{

c[6] = c[6] + quantity;

cout << "You have Selected :" << menu[6].menulist << endl;

break;

}

case 7:

{

c[7] = c[7] + quantity;

cout << "You have Selected :" << menu[7].menulist << endl;

break;

}

default:

cout << "invalid input" << endl;

}

cout << "to select more items (y/n)";

cin >> con;

} while (con != 'n');

cout << endl;

}

void cal()

{

double total = 0, tax, due;

cout << "------Welcome to Café international-----" << endl;

for (int i = 1; i < 8; i++)

{

if (c[i] > 0)

{

cout << c[i] << "\t" << menu[i].menulist << " RS " << menu[i].price << endl;

total = total + (menu[i].price\*c[i]);

}

}

tax = total\*0.17;

due = total + tax;

cout << " Tax " << "\t" << tax << endl;

cout << "-----------------------------------------------" << endl;

cout << "Amount due RS " << due << endl;

cout << "-----------------------------------------------" << endl;

}

**Question #06**

#include <iostream>

#include <cstdlib>

using namespace std;

void getData(int temp[12][2]){

for(int i=0;i<12;i++){

cout<<"Enter Teperatures of "<<i+1<<" Month"<<endl;

for(int j=0;j<2;j++){

cin>>temp[i][j];

}

}

}

void avH(int a[12][2]){

double totalt=0.0;

int ind=0,high=0;

for(int i=0;i<12;i++){

totalt+=a[i][0];

if(a[i][0]>high){

high=a[i][0];

ind=i;

}

}

cout<<"Average High Temperature= "<< totalt/12<<endl;

cout<<"Index of highest temperatur= "<<ind<<endl;

}

void avL(int a[12][2]){

double totalt=0.0;

int low=0,ind=0;

low=a[0][0];

for(int i=0;i<12;i++){

totalt+=a[i][1];

if(a[i][1]<low){

low=a[i][1];

ind=i;

}

}

cout<<"Average Low Temperatur= "<< totalt/12<<endl;

cout<<"Index of lowest temperatur= "<<ind<<endl;

}

int main(){

int temp[12][2];

getData(temp);

avH(temp);

avL(temp);

return 0;

}

**Question #07**

#include <iostream>

#include <fstream>

using namespace std;

struct BaseballID

{

string teamName, playerFirstName, playerLastName;

int homeRuns, rbi;

double batting\_average;

};

int main()

{

BaseballID listofplayers[10];

ifstream infile;

infile.open("/users/AlecKleyer/Desktop/computer science term 2/BaseballStats.txt");

if (!infile)

{

cout << "Error opening file!";

return 0;

}

for (int j = 0; j < 10; j++) {

infile >> listofplayers[j].teamName >> listofplayers[j].playerFirstName >> listofplayers[j].playerLastName >>listofplayers[j].homeRuns >> listofplayers[j].rbi >> listofplayers[j].batting\_average;

}

cout << "Please Type The Following Letter: ";

cout << "\n(A) For All Users and Stats";

cout << "\n(B) For A Specific Player";

cout << "\n(C) Print out for specific team";

cout << "\n(D) Update stats for a player";

char input = 0;

cin >> input;

if (input == 'A' || input == 'a') {

printInfoAll(\*listofplayers[]);

}

if (input == 'B' || input == 'b') {

}

if (input == 'C' || input == 'c') {

}

if (input == 'D' || input == 'd') {

}

}

void printInfoAll(listofplayers1[])

{

for (int i = 0; i < 10; i++) {

cout << &listofplayers[i];

}

}

**Chapter 12**

**Question #01**

#include <iostream>

#include <string>

using namespace std;

class romanType{

public:

romanType();

romanType(string s);

~romanType();

void setRoman(string);

int romanToDecimal();

void printDecimal();

void printRoman();

private:

string romanNum;

int decimalNum = 0;

};

romanType::romanType()

{

romanNum = 1;

}

romanType::~romanType()

{

}

void romanType::setRoman(string troll)

{

romanNum = troll;

}

int romanType::romanToDecimal()

{

for (int i = 0; i < romanNum.length(); i++)

{

if (romanNum[i] == 'I')

decimalNum++;

if (romanNum[i] == 'V')

{

if (i > 0 && romanNum[i - 1] == 'I')

decimalNum -= 2;

decimalNum += 5;

}

if (romanNum[i] == 'X')

{

if (i > 0 && romanNum[i - 1] == 'I')

decimalNum -= 2;

decimalNum += 10;

}

if (romanNum[i] == 'L')

{

if (i > 0 && romanNum[i - 1] == 'X')

decimalNum -= 20;

decimalNum += 50;

}

}

cout << decimalNum << endl;

return decimalNum;

}

int main()

{

string numerals;

do{

cout << "Enter roman numerals: ";

cin >> numerals;

romanType Rom;

Rom.setRoman(numerals);

Rom.romanToDecimal();

} while (true);

system("PAUSE");

}

**Question #02**

#include<iostream>

#include<string>

usingnamespace std;

void menu();

classdayType

{

string Wdays[7];

int i;

string day;

string prDay;

string NxtDay;

string AddDays;

public:

dayType(string);

string setday();

string preday();

void Nextday();

string add(int n);

void print();

};

int main()

{

int n;

string d;

menu();

cout <<"Enter the day :";

getline(cin, d);

dayType Da(d);

Da.setday();

Da.preday();

Da.Nextday();

cout <<"Enter the No. of days to add :";

while (!(cin >> n) ||n<0) {

cin.clear();

cin.ignore(999, '\n');

cout <<"Invalid data type! \nPlease enter No. of daysto add again :";

}

Da.add(n);

Da.print();

system("pause");

return 0;

}

dayType::dayType(stringi) :day(i){

Wdays[0] = "Mon";

Wdays[1] = "Tues";

Wdays[2] = "Wednes";

Wdays[3] = "Thurs";

Wdays[4] = "Fri";

Wdays[5] = "Satur";

Wdays[6] = "Sun";

}

stringdayType::setday()

{

if (day == Wdays[0])

{

i = 0;

}

elseif (day == Wdays[1])

{

i = 1;

}

elseif (day == Wdays[2])

{

i = 2;

}

elseif (day == Wdays[3])

{

i = 3;

}

elseif (day == Wdays[4])

{

i = 4;

}

elseif (day == Wdays[5])

{

i = 5;

}

elseif (day == Wdays[6])

i = 6;

else

{

day = "Invalid Input";

i = 7;

}

return day;

}

stringdayType::preday()

{

if (i == 0)

prDay = Wdays[6];

elseif (i == 7)

prDay = "Invalid Input";

else

prDay = Wdays[i - 1];

return prDay;

}

voiddayType::Nextday()

{

if (i == 6)

NxtDay = Wdays[0];

elseif (i == 7)

prDay = "Invalid Input";

else

NxtDay = Wdays[i + 1];

}

stringdayType::add(intn)

{

n = n + i;

while (n>= 7)

{

n = n - 7;

}

if (i == 7)

Wdays[n] = "Invalid Input ";

return AddDays = Wdays[n];

}

voiddayType::print()

{

cout << endl <<"\tDay="<< day <<"day"<< endl;

cout <<"\tPrevious day :"<< prDay <<"day"<< endl;

cout <<"\tNext day :"<< NxtDay <<"day"<< endl;

cout <<"\tAfter Adding Days :"<< AddDays <<"day"<< endl;

}

void menu()

{

cout <<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<< endl;

cout <<"\tEnter 'Sun' for 'Sunday'"<< endl;

cout <<"\tEnter 'Mon' for 'Monday'"<< endl;

cout <<"\tEnter 'Tues' for 'Tuesday'"<< endl;

cout <<"\tEnter 'Wednes' for 'Wednesday'"<< endl;

cout <<"\tEnter 'Thurs' for 'Thursday'"<< endl;

cout <<"\tEnter 'Fri' for 'Friday'"<< endl;

cout <<"\tEnter 'Satur' for 'Saturday'"<< endl;

}

**Question #06**

#include <iostream>

#include <string>

#include <iomanip>

using namespace std;

class bookType {

private:

string title;

string authors[4];

int n\_authors;

string ISBN;

string publisher;

int year;

double price;

int copies;

public:

bookType();

~bookType();

void set\_title(string str);

string get\_title();

void clear\_authors();

void add\_author(string str);

string get\_author(int number);

void set\_ISBN(string str);

string get\_ISBN();

void set\_publisher(string str);

string get\_publisher();

void set\_year(int n);

int get\_year();

void set\_price(double n);

double get\_price();

void set\_copies(int n);

int get\_copies();

void update\_copies(int n);

//! Actions.

void show\_title();

void show\_copies();

void show\_ISBN();

void show\_publisher();

void show\_price();

void show\_authors();

void show\_year();

//! 1 - equal, 0 - non-equal.

int compare\_title(string str);

};

bookType::bookType() {

title = "";

authors[0] = "";

authors[1] = "";

authors[2] = "";

authors[3] = "";

n\_authors = 0;

ISBN = "";

publisher = "";

year = 0;

price = 0;

copies = 0;

}

bookType::~bookType() {

}

void bookType::set\_title(string str) {

title = str;

}

string bookType::get\_title() {

return title;

}

void bookType::clear\_authors() {

n\_authors = 0;

}

void bookType::add\_author(string str) {

if (n\_authors < 4) {

authors[n\_authors] = str;

n\_authors++;

}

}

string bookType::get\_author(int number) {

if (n\_authors < number) {

return authors[number - 1];

}

else {

return "";

}

}

void bookType::set\_ISBN(string str) {

ISBN = str;

}

string bookType::get\_ISBN() {

return ISBN;

}

void bookType::set\_publisher(string str) {

publisher = str;

}

string bookType::get\_publisher() {

return publisher;

}

void bookType::set\_year(int n) {

year = n;

}

int bookType::get\_year() {

return year;

}

void bookType::set\_copies(int n) {

if (n >= 0) copies = n;

}

int bookType::get\_copies() {

return copies;

}

void bookType::set\_price(double n) {

if (n >= 0) price = n;

}

double bookType::get\_price() {

return price;

}

void bookType::update\_copies(int n) {

if (n >= 0) copies += n;

}

void bookType::show\_title() {

cout

<< "Title:"

<< title

<< endl;

}

void bookType::show\_copies() {

cout

<< "Copies in stock:"

<< copies

<< endl;

}

void bookType::show\_ISBN() {

cout

<< "ISBN:"

<< ISBN

<< endl;

}

void bookType::show\_publisher() {

cout

<< "Publisher:"

<< publisher

<< endl;

}

void bookType::show\_price() {

cout

<< "Price:"

<< setprecision(2)

<< fixed

<< price

<< endl;

}

void bookType::show\_year() {

cout

<< "Year of publish:"

<< year

<< endl;

}

void bookType::show\_authors() {

int i = 0;

cout

<< "Author(s):"

<< n\_authors

<< endl;

for(i = 0;i < n\_authors; i++) {

cout << authors[i] << " ";

}

cout << endl;

}

int bookType::compare\_title(string str) {

int x = (str == title);

return x;

}

int main()

{

bookType a[100];

int i;

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

a[i].add\_author("Newton");

a[i].add\_author("Kepler");

a[i].set\_copies(100-i);

a[i].set\_ISBN("123-123-123");

a[i].set\_price(100.01+i\*i);

a[i].set\_publisher("Piter");

a[i].set\_title("Star moving");

a[i].set\_year(1660+i);

}

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

a[i].show\_authors();

a[i].show\_copies();

a[i].show\_ISBN();

a[i].show\_price();

a[i].show\_publisher();

a[i].show\_title();

a[i].show\_year();

}

cout << "Compare(one) " << a[0].compare\_title("No one") << endl;

cout << "Compare(two) " << a[0].compare\_title("Star moving") << endl;

return 0;

}

**Question #07**

#include<iostream>  
#include<string>  
using namespace std;  
class memberType  
  
{  
public: string Name;  
string ID;  
int no\_books;  
float amount;  
memberType()  
{  
Name=" ";  
ID=" ";  
no\_books=0;  
amount=0.00;  
}  
  
public: void set(string name, string id, int no, float a);  
void modify(string name);  
void modify(int no, float a);  
void show();  
};  
  
void memberType::set(string n, string id, int no, float a)  
{  
Name=n;  
ID=id;  
no\_books=no;  
amount=a;  
}  
void memberType::modify(string name)  
{  
Name=name;  
}  
void memberType::modify(int no, float a)  
{  
no\_books=no;  
amount=a;  
}  
void memberType::show()  
{   
cout<<"Person's Name is: "<<Name<<endl;  
cout<<"Person's ID is: "<<ID<<endl;  
cout<<"Number of Books is: "<<no\_books<<endl;  
cout<<"Amount Spent is: "<<amount<<endl;  
}  
  
//-----------------------------------------------------------  
  
#include<iostream>  
#include<string>  
//#include<memberType.h>  
using namespace std;  
  
int main()  
{  
string n,id;  
int books;  
float a;  
memberType obj;  
  
cout<<"WELCOME TO PALOMA'S BOOK STORE"<<endl;  
cout<<"ENTER LIBRARY MEMBER'S NAME: ";  
cin>>n;  
cout<<"ENTER LIBRARY MEMBER'S ID: ";  
cin>>id;  
cout<<"ENTER NUMBER OF BOOKS BOUGHT: ";  
cin>>books;  
cout<<"ENTER DOLLAR AMOUNT PER BOOK: ";  
cin>>a;  
  
obj.set(n, id, books, a);  
obj.show();  
  
cout<<"ENTER UPDATED NAME: ";  
cin>>n;  
obj.modify(n);  
obj.show();  
  
cout<<"ENTER UPDATED BOOKS: ";  
cin>>books;  
  
obj.modify(books, a);  
obj.show();  
  
system("pause");  
}

**Question #10**

#include <iostream>

using namespace std;

int main()

{

double length, width, depth, rateFill, rateDrain;

cout << "Please enter the length: " << endl;

cin >> length;

cout << "Please enter the width: " << endl;

cin >> width;

cout << "Please enter the depth: " << endl;

cin >> depth;

cout << "Please enter the rate in which water fills the pool: " << endl;

cin >> rateFill;

cout << "Please enter the rate in which water drains the pool: " << endl;

cin >> rateDrain;

swimmingPool myPool(length, width, depth, rateFill, rateDrain);

double waterCapacity = myPool.poolTotalWaterCapacity();

cout << "Total water capacity: " << waterCapacity << endl;

double timeToFillPool = myPool.timeNeededToFill();

cout << "Time to fill the pool: " << timeToFillPool;

return 0;

}

swimmingPool::swimmingPool()

{

length = 0;

width = 0;

depth = 0;

rateWaterDrainsPool = 0;

rateWaterFillsPool = 0;

}

swimmingPool::swimmingPool(double l, double w, double d, double rateFill, double rateDrain)

{

length = l;

width = w;

depth = d;

rateWaterFillsPool = rateFill;

rateWaterDrainsPool = rateDrain;

}

double swimmingPool::amountWaterNeeded()

{

return (length \* width \* depth) \* GALLONS\_IN\_A\_CUBIC\_FOOT;

}

double swimmingPool::timeNeededToFill()

{

return poolTotalWaterCapacity() / rateWaterDrainsPool;

}

double swimmingPool::addWater(double time)

{

return rateWaterDrainsPool \* time;

}

double swimmingPool::drainWater(double time)

{

return rateWaterDrainsPool \* time;

}

double swimmingPool::poolTotalWaterCapacity()

{

return (length\*width\*depth)\*GALLONS\_IN\_A\_CUBIC\_FOOT;

}

class swimmingPool

{

public:

swimmingPool();

swimmingPool(double l, double w, double d, double rateFill, double rateDrain);

double amountWaterNeeded();

double timeNeededToFill();

double addWater(double time);

double drainWater(double time);

double poolTotalWaterCapacity();

private:

double length;

double width;

double depth;

double rateWaterFillsPool;

double rateWaterDrainsPool;

};

**Question #11**

#include <iostream>

#include <iomanip>

using namespace std;

class ticTacToe{

      char b[3][3]; //two dimenstional  array

public:

      ticTacToe(); //constructor

      void board(); //prints the game board

      void Input\_p1(int); //takes the input from the user

      bool Checker(); //checks the winner

};

int main()

{

      ticTacToe game; //creating object

      game.board(); //display the game board

      for (int i = 1; i <= 9; i++) //takes the game input from the user

      {

            game.Input\_p1(i);

            if (game.Checker())

                  break;

            if (i + 1 == 10)

            {

                  cout << "NO one WUNS\n";

            break;

            }

            i++;

            game.Input\_p1(i);

            if (game.Checker())

                  break;

            if (i+1 == 10)

                  cout << "NO one WUNS\n";

      }

      system("pause");

}

ticTacToe::ticTacToe() //constructor to intialize the 3by3 array

{

      b[0][0] = '1';

      b[0][1] = '2';

      b[0][2] = '3';

      b[1][0] = '4';

      b[1][1] = '5';

      b[1][2] = '6';

      b[2][0] = '7';

      b[2][1] = '8';

      b[2][2] = '9';

}

void ticTacToe::board()//function to create  the game board

{

      for (int i = 0; i < 3; i++)

            for (int j = 0; j < 3; j++)

            {

                  cout << " "<<b[i][j];

                  if (j <= 1)

                        cout << " |";

                  else

                        cout << endl;

                  if (j == 2&&i<2)

                        cout << "-----------" << endl;

            }

}

void ticTacToe::Input\_p1(int a) //function that takes input from the user

{

      int n, temp = 0;

      char c;

      if (a == 1 || a == 3 || a == 5 || a == 7 || a == 9)

      {

            cout << "Player 'X' Turn" << endl;

            c = 'X';

      }

      else

      {

            cout << "Player 'O' Turn" << endl;

            c = 'O';

      }

      cout << "Enter the Box no :";

      do{ // loop to take the box number AGAIN if user enters in correct number

            if (temp == 1)

                  cout << "  Again Enter the box no:";

            while (!(cin >> n))

            {

                  cin.clear();

                  cin.ignore(900, '\n');

                  cout << "  Invalid!\n  Again Enter the box no :";

            }

            if (n < 0 || n>9)

                  temp = 1;

            switch (n)

            {

            case 1:

                  if (b[0][0] == 'X'||b[0][0]=='O')

                        temp = 1;

                  else

                  {

                        b[0][0] = c;

                        temp = 0;

                  }

                  break;

            case 2:

                  if (b[0][1] == 'X' || b[0][1] == 'O')

                        temp = 1;

                  else

                  {

                        b[0][1] = c;

                        temp = 0;

                  }

                  break;

            case 3:

                  if (b[0][2] == 'X' || b[0][2] == 'O')

                        temp = 1;

                  else

                  {

                        b[0][2] = c;

                        temp = 0;

                  }

                  break;

            case 4:

                  if (b[1][0] == 'X' || b[1][0] == 'O')

                  temp = 1;

                  else

                  {

                        b[1][0] = c;

                        temp = 0;

                  }

                  break;

            case 5:

                  if (b[1][1] == 'X' || b[1][1] == 'O')

                        temp = 1;

                  else

                  {

                        b[1][1] = c;

                        temp = 0;

                  }

                  break;

            case 6:

                  if (b[1][2] == 'X' || b[1][2] == 'O')

                        temp = 1;

                  else

                  {

                        b[1][2] = c;

                        temp = 0;

                  }

                  break;

            case 7:

                  if (b[2][0] == 'X' || b[2][0] == 'O')

                        temp = 1;

                  else

                  {

                        b[2][0] = c;

                        temp = 0;

                  }

                  break;

            case 8:

                  if (b[2][1] == 'X' || b[2][1] == 'O')

                        temp = 1;

                  else

                  {

                        b[2][1] = c;

                      temp = 0;

                }

                  break;

            case 9:

                  if (b[2][2] == 'X' || b[2][2] == 'O')

                        temp = 1;

                  else

                  {

                        b[2][2] = c;

                        temp = 0;

                  }

                  break;

            }

      }while(temp != 0);

      board();

}

bool ticTacToe::Checker() //checks the winner

{

      if ((b[0][0] == 'X'&& b[0][1] == 'X'&&b[0][2] == 'X') ||

            (b[1][0] == 'X'&& b[1][1] == 'X'&&b[1][2] == 'X') ||

            (b[2][0] == 'X'&& b[2][1] == 'X'&&b[2][2] == 'X') ||

            (b[0][0] == 'X'&& b[1][0] == 'X'&&b[2][0] == 'X') ||

            (b[0][1] == 'X'&& b[1][1] == 'X'&&b[2][1] == 'X') ||

            (b[0][1] == 'X'&& b[1][1] == 'X'&&b[2][1] == 'X') ||

            (b[0][2] == 'X'&& b[1][1] == 'X'&&b[2][0] == 'X')||

            (b[0][0] == 'X'&& b[1][1] == 'X'&&b[2][2] == 'X'))

      {

            cout << "Player 'X' WINS" << endl;

            return 1;

      }

      else if ((b[0][0] == 'O'&& b[0][1] == 'O'&&b[0][2] == 'O') ||

            (b[1][0] == 'O'&& b[1][1] == 'O'&&b[1][2] == 'O') ||

            (b[2][0] == 'O'&& b[2][1] == 'O'&&b[2][2] == 'O') ||

            (b[0][0] == 'O'&& b[1][0] == 'O'&&b[2][0] == 'O') ||

            (b[0][1] == 'O'&& b[1][1] == 'O'&&b[2][1] == 'O') ||

            (b[0][1] == 'O'&& b[1][1] == 'O'&&b[2][1] == 'O') ||

            (b[0][2] == 'O'&& b[1][1] == 'O'&&b[2][0] == 'O') ||

            (b[0][0] == 'O'&& b[1][1] == 'O'&&b[2][2] == 'O'))

            cout << "Player 'O' WINS" << endl;

      else

            return 0;

}

**Chapter #13**

**Question #01**

#include <iostream>

#include <cstring>

#include "time.h"

using namespace std;

class clockType

{

public:

void getTime(int& hour, int& minute, int& second);

clockType(int myHr=12, int myMin=0, int mySec=0);

private:

int hr;

int min;

int sec;

};

class extClockType: public clockType

{

public:

void getTimeZone(string&);

extClockType(int hour=12, int min=0, int sec=0, string TZ="CST");

private:

string timeZ;

};

clockType::clockType(int myHr, int myMin, int mySec)

{

hr = myHr;

min = myMin;

sec = mySec;

};

extClockType::extClockType(int hour, int min, int sec, string TZ)

:clockType(int hour, int min, int sec)

{

timeZ = TZ;

};

void extClockType::getTimeZone(string& timeZone)

{

timeZone = timeZ;

};

int main(int argc, char \*argv[])

{

extClockType myClass(11, 30, 15, "CST");

cout << "The time zone is " << timeZ << endl;

system("PAUSE");

return EXIT\_SUCCESS;

}

**Question #3, 4 and 5**

#include<iostream>

using namespace std;

class pointType{

public:

int x;

int y;

pointType(){

x=0;

y=0;

}

pointType(int a,int b){

x=a;

y=b;

}

int set\_coordinate(int a, int b ){

x=a;

y=b;

}

void print(){

cout<<"X = "<<x<<endl;

cout<<"Y = "<<y<<endl;

}

int get\_X(){

return x;

}

int get\_y(){

return y;

}

};

class circle:public pointType{

private:

int rdius;

int center;

public:

circle():pointType(){

radius =0;

center =0;

}

circle(int x, int y, int r,int c):pointType(x,y){

radius =r;

center = c;

}

int set\_radius(int r){

radius=r;

}

int get\_center(int c){

center= c;

}

float area(){

float are= 3.14\*radius\*radius;

return are;

}

float circumference(){

float circum;

circum = 2\*3.14\*radius;

return circum;

}

void print(){

pointType::print();

cout<<"Radius = "<<radius<<endl;

cout<<"Center = "<<center<<endl;

}

};

class cylinder:public circle{

private:

int height;

public:

cylinder():circle(){

height=0;

}

cylinder(int x, int y, int r,int c,int h):circle(x,y,r,c){

height=h;

}

int set\_height(int h){

height= h;

}

int get\_height(){

return height;

}

int volume(){

int vol;

vol= 3.14\*radius\*radius\*height;

}

int surfaceArea(){

int sur;

sur= (2\*3\*radius\*height) + (2\*3\*radius\*radius);

}

print(){

circle::print();

cout<<"Volume = "<<volume<<endl;

cout<<"Surface Area = "<<surfaceArea<<endl;

}

};

int main(){

cylinder obj(5,4,6,7,8);

obj.print();

cout<<"Volume = "<<obj.area()<<endl;

cout<<"Surface Area = "<<obj.circumference()<<endl;

}

**Question #06**

#include<iostream>

#include<string>

using namespace std;

class dateType

{

int month, day, year;

public:

dateType(int m=1, int d=1, int y=1900);

int getDay();

int getMonth();

int getYear();

void printDate();

};

xtPersonType::extPersonType()

{

}

extPersonType::extPersonType(extPersonType person, dateType dob, addressType address, string ph, int type)

{

Person=person;

DOB=dob;

Address=address;

Type=type;

phonenumber=ph;

}

void extPersonType::setType(int type)

{

if(type>=1 && type<=3)

Type = type;

else

Type=1;

}

string extPersonType::getType()

{

if(Type==1)

return "Family Member";

else if (Type==2)

return "Friend";

else if(Type==3)

return "Business Associate";

}

void extPersonType::printPersonDetails()

{

Person.print();

Address.printAddress();

DOB.printDate();

cout<<getType();

cout<<endl<<"Phone: "<<phonenumber<<endl;

}

void extPersonType::setPhone(string ph)

{

phonenumber=ph;

}

string extPersonType::getPhone()

{

return phonenumber;

}

void extPersonType::setPerson(personType person)

{

Person=person;

}

void extPersonType::setDOB(dateType dob)

{

DOB=dob;

}

void extPersonType::setAddress(addressType address)

{

Address=address;

}

personType extPersonType::getPerson()

{

return Person;

}

dateType extPersonType::getDate()

{

return DOB;

}

int extPersonType::getIntType()

{

return Type;

}

addressType extPersonType::getAddress()

{

return Address;

}

**Question #07**

#include <iostream>

using namespace std;

class Date{

private:

int month;

int day;

int year;

public:

Date();

Date(int month,int day,int year);

void print\_f1();

void print\_f2();

void print\_f3();

};

int main()

{

Date myDate;

myDate.print\_f1();

cout<<endl;

myDate.print\_f2();

cout<<endl;

myDate.print\_f3();

cout<<endl;

system("pause");

return 0;

}

//class implementation

Date::Date()

{

cout << "Enter the number for the month.";

cin >> month;

cout << "Enter the day.";

cin >> day;

cout << "Enter a four digit year.";

cin >> year;

}

Date::Date(int month,int day,int year)

{

if((month > 1) || (month > 12) || (day <1) || (day > 31) || (year < 1950) || (year > 2020))

cout << "Illegal data." << endl;

}

void Date::print\_f1()

{

string Month;

switch(month){//class data member: month

case 1:

Month="January";

break;

case 2:

Month="February";

break;

case 3:

Month="March";

break;

case 4:

Month="April";

break;

case 5:

Month="May";

break;

case 6:

Month="June";

break;

case 7:

Month="July";

break;

case 8:

Month="August";

break;

case 9:

Month="September";

break;

case 10:

Month="October";

break;

case 11:

Month="November";

break;

case 12:

Month="December";

break;

}

cout << Month << '/'<< day << '/' <<year;

}

void Date::print\_f2()

{

cout << month << '/' << day << '/' << year;

}

**Question #13**

#include <iostream>

using namespace std;

class bankAccount

{

public:

bankAccount();

bankAccount(int, double, double, double);

~bankAccount(){}

void setAccountNum(int);

int getAccountNum() const;

double getBalance() const;

void deposit(double);

void withdraw(double);

void printAccountInfo() const;

void setBalance(double);

private:

int accountNum;

protected:

double accountBalance;

};

bankAccount::bankAccount()

{

accountNum = 0;

accountBalance = 0;

};

bankAccount::bankAccount(int n, double b, double d, double w)

{

setAccountNum(n);

setBalance(b);

deposit(d);

withdraw(w);

};

void bankAccount::setAccountNum(int n)

{

accountNum = n;

};

int bankAccount::getAccountNum() const

{

return accountNum;

};

void bankAccount::setBalance(double b)

{

accountBalance = b;

}

double bankAccount::getBalance() const

{

return accountBalance;

};

void bankAccount::deposit(double d)

{

if (d >= 0)

{

accountBalance += d;

}

else

{

cout << "The withdraw ammount is invalid. The withdraw ammount will be set to zero." << endl;

d = 0;

accountBalance += d;

}

};

void bankAccount::withdraw(double w)

{

if (w >= 0)

{

accountBalance -= w;

}

else

{

cout << "The withdraw ammount is invalid. The withdraw ammount will be set to zero." << endl;

w = 0;

accountBalance -= w;

}

};

void bankAccount::printAccountInfo() const

{

cout << "Account Number: " << accountNum << endl;

cout << "Account Balance: " << accountBalance << endl;

};

using namespace std;

class checkingAccount : public bankAccount

{

public:

checkingAccount();

~checkingAccount(){}

checkingAccount(int, double, double, double, double, double, double);

void setInterest(double);

void setminBalance(double);

double getminBalance();

void setServiceCharges(double);

double getServiceCharges();

void postInterest();

void withdraw(double);

void printAccountInfo() const;

void CheckAmount(double);

private:

double interest;

double minBalance;

double serviceCharges;

};

checkingAccount::checkingAccount() : bankAccount()

{

interest = 0;

minBalance = 0;

serviceCharges = 0;

}

checkingAccount::checkingAccount(int n, double b, double d, double w, double i, double mb, double sc) : bankAccount(n, b, d, w)

{

setInterest(i);

setminBalance(mb);

setServiceCharges(sc);

}

void checkingAccount::setInterest(double i)

{

interest = i;

}

void checkingAccount::setminBalance(double mb)

{

minBalance = mb;

}

double checkingAccount::getminBalance()

{

return minBalance;

}

void checkingAccount::setServiceCharges(double sc)

{

serviceCharges = sc;

}

double checkingAccount::getServiceCharges()

{

return serviceCharges;

}

void checkingAccount::postInterest()

{

accountBalance \*= (1 + interest / 100);

}

void checkingAccount::CheckAmount(double c)

{

accountBalance -= c;

}

void checkingAccount::printAccountInfo() const

{

cout << "Account Number: " << getAccountNum() << endl;

cout << "Account Balance: " << getBalance() << endl;

cout << "Minimum Balance: " << minBalance << endl;

cout << "Interest: " << interest << endl;

cout << "Service Charge: " << serviceCharges << endl;

};

void checkingAccount::withdraw(double w)

{

accountBalance -= w;

if ((accountBalance) < minBalance)

{

accountBalance -=serviceCharges;

}

}

int main()

{

checkingAccount Customer1;

Customer1.setAccountNum(100584220);

Customer1.setBalance(2500);

Customer1.setServiceCharges(15);

Customer1.setminBalance(350);

Customer1.setInterest(5);

Customer1.deposit(250);

Customer1.withdraw(159.55);

Customer1.CheckAmount(1000);

Customer1.postInterest();

Customer1.printAccountInfo();

system("PAUSE");

return 0;

}