ASSIGNMENT NO 02

**Q#1: Solve chapter# 4,5,6 exercise Questions (minimum 10 questions), make a word document of your assignment for submission in hard form, upload your assignment at GitHub profile and mention your GitHub ID in the word file.**

# **CHAPTER # 4**

## **PROGRAM NO. 4.13**: (Gas Mileage) Drivers are concerned with the mileage obtained by their automobiles. One driver has kept track of several trips by recording miles driven and gallons used for each trip. Develop a C++ program that uses a while statement to input the miles driven and gallons used for each trip. The program should calculate and display the miles per gallon obtained for each trip and print the combined miles per gallon obtained for all tankfuls up to this point.

#include<iostream>

#include<string>

using namespace std;

int main(){

int km;

int liter;

cout<<"Enter kilometers driven (-1 to quit): ";

cin>>km;

float avg = 0;

int count = 0;

while (km!= -1){

count++;

cout<<"Enter liters used: ";

cin>>liter;

float curr= km/liter;

avg =curr/count;

cout<<"Kms per liter this trip: " << curr <<"\n";

cout<<"Total kms per liter: " << avg <<"\n";

cout<<"\n";

cout<<"Enter kilometers driven (-1 to quit): ";

cin>>km;

}

}

**PROGRAM NO.4.14**: (Credit Limits) Develop a C++ program that will determine whether a department-store customer has exceeded the credit limit on a charge account. For each customer, the following facts are available: a) Account number (an integer) b) Balance at the beginning of the month c) Total of all items charged by this customer this month d) Total of all credits applied to this customer's account this month e) Allowed credit limit The program should use a while statement to input each of these facts, calculate the new balance (= beginning balance + charges – credits) and determine whether the new balance exceeds the customer’s credit limit. For those customers whose credit limit is exceeded, the program should display the customer’s account number, credit limit, new balance and the message “Credit Limit Exceeded.”

#include <iostream>

#include <string>

using namespace std;

int main(){

int account\_number;

float balance,total\_charges,total\_credit,credit\_limit,new\_balance;

cout<<"Enter Accoutn Number(or -1 to quit):";

cin>>account\_number;

while (account\_number!=-1)

cout<<"Enter beginning Balance:";

cin>>balance;

cout<<"Enter Total Charges:";

cin>>total\_charges;

cout<<"Enter Total Credits:";

cin>>total\_credit;

cout<<"Enter Credit Limit:";

cin>>credit\_limit;

new\_balance=(balance+total\_charges)-total\_credit;

cout<<"New Balance is:"<<new\_balance<<"\n";

cout<<"Accoutn:"<<account\_number<<"\n";

cout<<"Credit limit:"<<credit\_limit<<"\n";

cout<<"Balance:"<<new\_balance<<"\n";

if(credit\_limit<new\_balance){

cout<<"Credit Limit Exceeded!\n";

}

cout<<"\n\n";

cout<<"Enter Accoutn Number(or -1 to quit):";

cin>>account\_number;

}

return 0;

}

**PROGRAM NO.4.16:** (Salary Calculator) Develop a C++ program that uses a while statement to determine the gross pay for each of several employees. The company pays “straight time” for the first 40 hours worked by each employee and pays “time-and-a-half” for all hours worked in excess of 40 hours. You are given a list of the employees of the company, the number of hours each employee worked last week and the hourly rate of each employee. Your program should input this information for each employee and should determine and display the employee’s gross pay.

#include<iostream>

#include<string>

using namespace std;

int main() {

double total\_funds = 0;

while (true) {

int laps;

double sponsorship\_rate, contribution;

cout << "Enter laps completed(-1 to end): ";

cin >> laps;

if (laps == -1) {

cout << "Total funds raised: " << total\_funds;

return 0;

}

cout << "Enter sponsorship rate: ";

cin >> sponsorship\_rate;

contribution = laps\*sponsorship\_rate;

if (laps > 40) {

contribution += (40-laps)\*sponsorship\_rate\*1.5;

}

cout << "Student contribution: " << contribution << "\n";

total\_funds += contribution; }

return 0; }

**PROGRAM NO. 4.17:** (Find the Largest) The process of finding the largest number (i.e., the maximum of a group of numbers) is used frequently in computer applications. For example, a program that determines the winner of a sales contest inputs the number of units sold by each salesperson. The salesperson who sells the most units wins the contest. Write a C++ program that uses a while statement to determine and print the largest number of 10 numbers input by the user. Your program should use three variables, as follows: counter: A counter to count to 10 (i.e., to keep track of how many numbers have been input and to determine when all 10 numbers have been processed). number: The current number input to the program. largest: The largest number found so far.

#include <iostream>

#include <string>

using namespace std;

int main(){

int counter=1;

int sahil;

int units;

int secondmax=0,max=0;

while (counter<=10) {

cout<<"Please Enter Units for Salesman # "<<counter<<" ";

cin>>units;

if(max<units){

secondmax=max;

max=units;

sahil=counter;

}

counter++;

}

cout<<"The Maximum Units Selled are:"<<max<<endl;

cout<<"The Maximum Units Selled by Saleman no:"<<sahil<<endl;

return 0;

}

**PROGRAM NO. 4.19:** (Find the Two Largest Numbers) Using an approach similar to that in Exercise 4.17, find the two largest values among the 10 numbers.

#include <iostream>

using namespace std;

int main() {

int counter{ 0 };

int largest1{ 0 };

int largest2{ 0 };

while ( ++counter <= 10 ) {

int number;

cout << "Number " << counter << ": ";

cin >> number;

if ( number > largest1 ) {

largest1 = number;

if ( largest1 > largest2 ) {

number = largest2;

largest2 = largest1;

largest1 = number;

}}

}

cout << "\nLargest numbers are " << largest1 << " and " << largest2 << endl;

return 0;

}

**PROGRAM NO. 4.20:** (Validating User Input) The examination-results program of Fig. 4.16 assumes that any value input by the user that’s not a 1 must be a 2. Modify the application to validate its inputs. On any input, if the value entered is other than 1 or 2, keep looping until the user enters a correct value.

#include <iostream>

using namespace std;

int main() {

unsigned int passes{ 0 };

unsigned int failures{ 0 };

unsigned int studentCounter{ 1 };

while ( studentCounter <= 10 ) {

cout << "Enter result (1 = pass, 2 = fail): ";

int result;

cin >> result;

if ( result == 1 ) {

passes++;

studentCounter++;

}

else if ( result == 2 ) {

failures++;

studentCounter++;

}

else

cout << "Invalid result!\n";

}

cout << "Passed: " << passes << "\nFailed: " << failures << endl;

if ( passes > 8 )

cout << "Bonus to instructor!" << endl;

}

**PROGRAM NO. 4.26:** (Palindromes) A palindrome is a number or a text phrase that reads the same backward as forward. For example, each of the following five-digit integers is a palindrome: 12321, 55555, 45554 and 11611. Write a program that reads in a five-digit integer and determines whether it’s a palindrome.

#include <iostream>

using namespace std;

int main() {

int number{ 0 };

while ( !number ) {

cout << "Five-digit number: ";

cin >> number;

if ( number < 10000 )

number = 0;

if ( number > 99999 )

number = 0;

}

if ( number / 10000 == number % 10 )

if ( number % 10000 / 1000 == number % 100 / 10 )

cout << "Palindrome!\n";

return 0;

}

**PROGRAM NO. 4.33:** (Sides of a Right Triangle) Write a program that reads three nonzero integers and determines and prints whether they’re the sides of a right triangle.

#include <iostream>

using namespace std;

int main() {

int a, b, c;

cout << "Three nonzero integers: ";

cin >> a >> b >> c;

if ( a \* a == b \* b + c \* c )

cout << "Right triangle!";

else if ( b \* b == a \* a + c \* c )

cout << "Right triangle!";

else if ( c \* c == a \* a + b \* b )

cout << "Right triangle!";

cout << endl;

return 0;

}

# **CHAPTER # 5**

**PROGRAM NO. 5.11:** (Compound Interest) Modify the compound interest program of Section 5.4 to repeat its steps for the interest rates 5%, 6%, 7%, 8%, 9% and 10%. Use a for statement to vary the interest rate.

#include <iostream>

using namespace std;

int main() {

int number;

int smallest;

cout << "Number of values: ";

cin >> number;

cout << "\nValue: ";

cin >> smallest;

while ( --number ) {

int value;

cout << "Value: ";

cin >> value;

if ( value < smallest )

smallest = value;

}

cout << "\nSmallest is " << smallest << endl;

return 0;

}

**PROGRAM NO. 5.13:** (Bar Chart) One interesting application of computers is drawing graphs and bar charts. Write a program that reads five numbers (each between 1 and 30). Assume that the user enters only valid values. For each number that is read, your program should print a line containing that number of adjacent asterisks. For example, if your program reads the number 7, it should print \*\*\*\*\*\*\*.

#include <iostream>

using namespace std;

void printLine(int);

void printLine(int value) {

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

cout << "\*";

}

cout << endl;

}

int main() {

int counter = 5;

int value = 0;

cout << "Program to print bar chart of entered values.\n" << std::endl;

while (counter > 0) {

printf("Enter an integer value 1-30 (%d remaining): ", counter);

cin >> value;

if (value >= 1 && value <= 30) {

printLine(value);

counter--;

} else {

cout << "Incorrect value entered." << std::endl}

}return 0;

}

**PROGRAM NO. 5.14:** (Calculating Total Sales) A mail order house sells five different products whose retail prices are: product 1 — $2.98, product 2—$4.50, product 3—$9.98, product 4—$4.49 and product 5— $6.87. Write a program that reads a series of pairs of numbers as follows: a) product number b) quantity sold Your program should use a switch statement to determine the retail price for each product. Your program should calculate and display the total retail value of all products sold. Use a sentinel-controlled loop to determine when the program should stop looping and display the final results.

#include <iostream>

#include <string>

using namespace std;

double getPrice(int, int);

int main() {

double total = 0.0f;

int product = 0, quantity = 0;

cout << "Program to add retail prices for 5 products.\n" <<endl;

while (product != -1) {

cout<< "Enter the product number and quantity sold (-1 to quit): ";

cin >> product;

if (product != -1) {

cin >> quantity;

total += getPrice(product, quantity);

printf("Running total: $%.2f\n", total);

}

}

return 0;

}

//Outise Main Function

double getPrice(int product, int quantity) {

switch (product) {

case 1:

return 2.98 \* quantity;

break;

case 2:

return 4.50 \* quantity;

break;

case 3:

return 9.98 \* quantity;

break;

case 4:

return 4.49 \* quantity;

break;

case 5:

return 6.87 \* quantity;

break;

default:

cout << "Incorrect product number entered." <<endl;

return 0.0f;

break;

}**}**

**PROGRAM NO. 5.15:** (GradeBook Modification) Modify the GradeBook program of Figs. 5.9–5.11 to calculate the grade-point average. A grade of A is worth 4 points, B is worth 3 points, and so on.

#include<iostream>

using namespace std;

int main()

{

for(int i = 10; i >= 1; i--)

{

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

{

cout << "\* ";

}

cout << "\n";

}

return 0;

}

**PROGRAM NO. 5.19:** (Calculating π) Calculate the value of π from the infinite series Print a table that shows the approximate value of π after each of the first 1000 terms of this series.

#include <iostream>

using namespace std;

int main() {

double pi{ 0.0 };

cout << "Step\tPi\n" << endl;

for ( int i{ 1 }; i <= 20; ++i ) {

double term{ 4.0 / ( i \* 2 - 1 ) };

i % 2 ? pi += term : pi -= term;

cout << i << '\t' << pi << endl; }

return 0;

}

**PROGRAM NO. 5.21:** (Calculating Salaries) A company pays its employees as managers (who receive a fixed weekly salary), hourly workers (who receive a fixed hourly wage for up to the first 40 hours they work and “time-and-a-half”—1.5 times their hourly wage—for overtime hours worked), commission workers (who receive $250 plus 5.7 percent of their gross weekly sales), or pieceworkers (who receive a fixed amount of money per item for each of the items they produce—each pieceworker in this company works on only one type of item). Write a program to computethe weekly pay foreach employee. You do not know the number of employees in advance. Each type ofemployee has its own pay code: Managers have code 1, hourly workers have code 2, commission workers have code 3 and pieceworkers have code 4. Use a switch to compute each employee’s pay according to that employee’s paycode. Within the switch, prompt the user (i.e., the payroll clerk) to enter the appropriate facts your program needs to calculate each employee’s pay according to that employee’s paycode.

#include <iostream>

using namespace std;

int main() {

for ( int row{ 1 }; row <= 10; ++row ) {

for ( int col{ 1 }; col <= row; ++col )

cout << '\*';

for ( int col{ 10 }; col > row; --col )

cout << ' ';

cout << '\t';

for ( int col{ 10 }; col >= row; --col )

cout << '\*';

for ( int col{ 1 }; col < row; ++col )

cout << ' ';

cout << '\t';

for ( int col{ 1 }; col < row; ++col )

cout << ' ';

for ( int col{ 10 }; col >= row; --col )

cout << '\*';

cout << '\t';

for ( int col{ 10 }; col > row; --col )

cout << ' ';

for ( int col{ 1 }; col <= row; ++col )

cout << '\*';

cout << endl;

}

return 0;

}

**PROGRAM NO. 5.22:** (De Morgan’s Laws) In this chapter, we discussed the logical operators &&, || and !. De Morgan’s laws can sometimes make it more convenient for us to express a logical expression. These laws state that the expression !( condition1 && condition2 ) is logically equivalent to the expression ( !condition1 || !condition2 ). Also, the expression !( condition1 || condition2 ) is logically equivalent to the expression ( !condition1 && !condition2 ). Use De Morgan’s laws to write equivalent expressions for each of the following, then write a program to show that the original expression and the new expression in each case are equivalent:

a) !( x < 5 ) && !( y >= 7 )

b) !( a == b ) || !( g != 5 )

c) !( ( x <= 8 ) && ( y > 4 ) )

d) !( ( i > 4 ) || ( j <= 6 ) )

#include <iostream>

using namespace std;

int main() {

int x{ 1 };

int y{ 2 };

int a{ 3 };

int b{ 4 };

//x = y = a = b = 0; // all-to-0 test

if ( !( x <= 6 && y % 2 == 1 ) == ( !( x <= 6 ) || !( y % 2 == 1 ) ) )

cout << true;

if ( !( a < 4 || b >= 6 ) == ( !( a < 4 ) && !( b >= 6 ) ) )

cout << true;

if ( ( !( x < 3 ) && !( y >= 2 ) ) == !( x < 3 || y >= 2 ) )

cout << true;

if ( ( !( a == b ) || !( b != 2 ) ) == !( a == b && b != 2 ) )

cout << true;

cout << endl;

return 0;

**PROGRAM NO.5.23:** (Diamond of Asterisks) Write a program that prints the following diamond shape. You may use output statements that print a single asterisk (\*), a single blank or a single newline. Maximize your use of repetition (with nested for statements) and minimize the number of output statements.

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

#include <iostream>

using namespace std;

int main() {

int ast{ 9 }; // asterisks

for ( int y{ 1 }; y <= 9; ++y ) {

for ( int x{ 1 }; x <= 9; ++x ) {

if ( x == 1 || y == 1 || x == 9 || y == 9 )

cout << '#';

else {

for ( int x{ 1 }; x < ( 9 - ast ) / 2; ++x )

cout << ' ';

for ( int x{ 1 }; x <= ast; ++x )

cout << '\*';

for ( int x{ 1 }; x < ( 9 - ast ) / 2; ++x )

cout << ' ';

x = 8;

}

}

y < 5 ? ast -= 2 : ast += 2;

cout << '\n';

}

return 0;

**PROGRAM NO.5.27:** (Removing the continue Statement) Describe in general how you’d remove any continue statement from a loop in a program and replace it with some structured equivalent. Use the technique you developed here to remove the continue statement from the program of Fig. 5.14.

#include <iostream>

using namespace std;

int main() {

for ( unsigned int count{ 1 }; count <= 10; count++ ) {

if ( count == 5 )

++count;

cout << count << ' ';

}

cout << "\nUsed increment to skip printing 5" << endl;

return 0;

}

**CHAPTER # 6**

**PROOGRAM NO. 6.13:** (Rounding Numbers) An application of function floor is rounding a value to the nearest integer. The statement

y = floor( x + 0.5 );

rounds the number x to the nearest integer and assigns the result to y. Write a program that reads several numbers and uses the preceding statement to round each of these numbers to the nearest integer. For each number processed, print both the original number and the rounded number.

#include <cmath>

#include <iostream>

using namespace std;

int main() {

for ( int i{ 1 }; i <= 5; ++i ) {

double x;

cout << "Number: ";

cin >> x;

double y{ floor( x + 0.5 ) };

cout << "Nearest integer for " << x << " is " << y << '\n' << endl;

}

return 0;

}

**PROGRAM NO. 6.17:** (Random Numbers) Write a single statement that prints a number at random from each of the following sets: a) 2, 4, 6, 8, 10. b) 3, 5, 7, 9, 11. c) 6, 10, 14, 18, 22.

#include <cstdlib>

#include <ctime>

#include <iostream>

using namespace std;

int main() {

srand( time( 0 ) );

cout << 0 + rand() % 5 \* 3 << endl; // a

cout << 3 + rand() % 6 \* 2 << endl; // b

cout << 6 + rand() % 4 \* 4 << endl; // c

return 0;

}

**PROGRAM NO.6.18:** (Exponentiation) Write a function integerPower(base, exponent) that returns the value of base exponent For example, integerPower(3, 4) = 3 \* 3 \* 3 \* 3. Assume that exponent is a positive, nonzero integer and that base is an integer. Do not use any math library functions.

#include <iostream>

using namespace std;

long integerPower( int, unsigned );

int main() {

int b;

unsigned e;

cout << "Base and exponent: ";

cin >> b >> e;

cout << "Result is " << integerPower( b, e ) << endl;

return 0;

}

long integerPower( int base, unsigned exponent ) {

long value{ 1 };

do value \*= base;

while ( --exponent );

return value;

}

**PROGRAM NO.6.19:** (Hypotenuse Calculations) Define a function hypotenuse that calculates the hypotenuse of a right triangle when the other two sides are given. The function should take two double arguments and return the hypotenuse as a double. Use this function in a program to determine the hypotenuse for each of the triangles shown below.

**Triangle Side 1 Side 2**

1 3.0 4.0

2 5.0 12.0

3 8.0 15.0

#include <cmath>

#include <iostream>

using namespace std;

double hypotenuse( double, double );

int main() {

cout << "1: " << hypotenuse( 3.0, 4.0 )

<< "\n2: " << hypotenuse( 5.0, 12.0 )

<< "\n3: " << hypotenuse( 8.0, 15.0 )

<< endl;

return 0;

}

double hypotenuse( double s1, double s2 ) {

return sqrt( s1 \* s1 + s2 \* s2 );

}

**PROGRAM NO.6.20:** (Multiples) Write a function multiple that determines for a pair of integers whether thesecond is a multiple of the first. The function should take two integer arguments and return true if the second is a multiple of the first, false otherwise. Use this function in a program that inputs a series of pairs of integers.

#include <iostream>

using namespace std;

bool isFactor( int, int );

int main() {

while ( true ) {

int a, b;

cout << "Two integer numbers (0 to end): ";

cin >> a;

if ( !a ) break;

cin >> b;

cout << b << " is " << ( isFactor( a, b ) ? "" : "not " )

<< "a factor of " << a << "\n\n";

}

return 0;

}

bool isFactor( int n1, int n2 ) {

return !( n1 % n2 );

}

**PROGRAM NO.6.21:** (Even Numbers) Write a program that inputs a series of integers and passes them one at a time to function isEven, which uses the modulus operator to determine whether an integer is even. The function should take an integer argument and return true if the integer is even and false otherwise

#include <iostream>

using namespace std;

bool isMultiple3( int );

int main() {

while ( true ) {

int n;

cout << "Integer number (0 to end): ";

cin >> n;

if ( !n ) break;

cout << n << " is " << ( isMultiple3( n ) ? "" : "not " )

<< "a multiple of 3\n\n";

}

return 0;

}

bool isMultiple3( int number ) {

return !( number % 3 );

}

**PROGRAM NO.6.22:** (Square of Asterisks) Write a function that displays at the left margin of the screen a solid square of asterisks whose side is specified in integer parameter side. For example, if side is 4, the function displays the following.

\*\*\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

#include <iostream>

using namespace std;

void displayRectangle( int, int );

int main() {

int w, h;

do {

cout << "Width and height: ";

cin >> w >> h;

} while ( w < 1 || h < 1 );

cout << endl;

displayRectangle( w, h );

return 0;

}

void displayRectangle( int width, int height ) {

for ( int y{ 1 }; y <= height; ++y ) {

for ( int x{ 1 }; x <= width; ++x )

cout << '\*';

cout << endl;

}

}

**PROGRAM NO.6.27:** (Find the Minimum) Write a program that inputs three double-precision, floating-point numbers and passes them to a function that returns the smallest number.

#include <iostream>

using namespace std;

double smallest( double, double, double );

int main() {

double d1, d2, d3;

cout << "Three double numbers: ";

cin >> d1 >> d2 >> d3;

cout << "Smallest is " << smallest( d1, d2, d3 ) << endl;

return 0;

}

double smallest( double min, double n2, double n3 ) {

if ( n2 < min ) min = n2;

return n3 < min ? n3 : min;

}

**PROGRAM NO.6.29**(Prime Numbers) An integer is said to be prime if it’s divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime, but 4, 6, 8 and 9 are not.

a) Write a function that determines whether a number is prime.

b) Use this function in a program that determines and prints all the prime numbers between 2 and 10,000. How many of these numbers do you really have to test before being sure that you’ve found all the primes?

c) Initially, you might think that n/2 is the upper limit for which you must test to see whether a number is prime, but you need only go as high as the square root of n. Why? Rewrite the program, and run it both ways. Estimate the performance improvement

#include <cmath>

#include <iostream>

using namespace std;

bool isPrime( unsigned );

int main() {

cout << "Prime numbers:\n" << endl;

for ( unsigned n{ 2 }; n <= 10'000; ++n )

if ( isPrime( n ) )

cout << n << ' ';

cout << endl;

return 0;

}

bool isPrime( unsigned number ) {

for ( unsigned i{ 2 }; i <= sqrt( number ); ++i )

if ( !( number % i ) )

return false;

return true;

}

**PROGRAM NO. 6.30:** (Reverse Digits) Write a function that takes an integer value and returns the number with its digits reversed. For example, given the number 7631, the function should return 1367.

#include <iostream>

using namespace std;

unsigned reverse( unsigned );

int main() {

unsigned n;

cout << "Number: ";

cin >> n;

cout << "Reversed is " << reverse( n ) << endl;

return 0;

}

unsigned reverse( unsigned number ) {

unsigned div{ 1 };

while ( number / div )

div \*= 10;

unsigned reversed{ 0 };

while ( number ) {

div /= 10;

reversed += number % 10 \* div;

number /= 10;

}

return reversed;

}

**Q#2 : Solve “Case study” Integer Based Monetary Calculations with Class Dollar Amount” (5.7 Page211) with balanced functionality mentioned in exercise questions(5.30,5.31,5.33 ,5.34,5.35).**

**QUESTION 5.30**

|  |
| --- |
| #include <iostream> |
|  | #include <iomanip> |
|  | #include <cmath> // standard C++ math library |
|  | using namespace std; |
|  |  |
|  | int main() |
|  | { |
|  | double amount; // amount on deposit at end of each year |
|  | double principal = 24.0; // initial amount before interest |
|  | double rate = .05; // interest rate |
|  |  |
|  | // display headers |
|  | cout << "Year" << setw( 24 ) << "Amount on deposit" << endl; |
|  |  |
|  | // set floating-point number format |
|  | cout << fixed << setprecision( 2 ); |
|  | for ( rate = .05; rate <= 0.1; rate += 0.01) |
|  | { |
|  | // calculate amount on deposit for each of ten years |
|  | for ( int year = 1; year <= 384; ++year ) |
|  | { |
|  | // calculate new amount for specified year |
|  | amount = principal \* pow( 1.0 + rate, year ); |
|  |  |
|  | // display the year and the amount |
|  | cout << setw( 4 ) << year << setw( 24 ) << amount << endl; |
|  | } // end for |
|  | cout << endl; |
|  | } |
|  | } // end main |

**Question 5.31**#include <iostream>

using namespace std;

void Cal\_Print(int age[], int INCOME[]){

int tax;

if (age<55&&INcome <= 10000){

tax = 0;

}

else if (age<55 && INcome >10000 && INcome<=50000){

tax = INcome\*10/100;

}

else if (age<55 &&INcome >50000 && INcome <=100000){

tax =INcome\*20/100;

}

else if (age<55&&INcome >100000){

tax = INcome\*30/100;

else if (age>=55 && age<80&&INcome <= 20000){

tax = 0;

}

else if (age>=55&&age<80&&INcome >20000 && INcome<=70000){

tax = INcome\*10/100;

}

else if (age>=55&&age<80&&INcome >70000 && INcome<=150000){

tax =INcome\*20/100;

}

else if (age>=55&&age<80&&INcome >150000){

tax = INcome\*30/100;

}

else if (age >= 80){

tax = 0;

}

}

int main()

{

int fnpfNbr[6] = {12345, 12361, 34763, 11224, 54129, 10717};

int size=6;

int Age[6] = {24, 19, 47, 50, 35, 26};

char lastname\_initial[6] = {'F', 'B', 'H', 'H', 'R', 'B'};

int Income[] = {40000, 20000, 100000, 35000, 75000, 28000};

int Tax[6];

Cal\_Print(Age[], Income[]);

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

cout <<lastname\_initial[i]<<"\t"<<fnpfNbr[i]<<"\t"<<Age[i]<<"\t"<<Income[i]<<"\t"<<Tax[i]<<"\n";

}

return 0;

}

**Q#3 : We have to tell about 2ND Semester Project idea plan what we wants to do in OOP (OBJECT ORIENTED PARADIGM).** **Project:**

**Name : Chess game or Chase Robot game**

**Members Name:**

**Muhammad Bilal.**

**Muhammad Osama.**

**Aseel Malik.**