**Lab 17**

# Task 1

#include<iostream>

#include<string>

int main()

{

std::string str,reverse; //Taking Inputs

std::cout << "Enter a string ";

std::getline(std::cin, str);

int count = 0;

for (int i = 0; str[i] != '\0'; i++)

{

count++;

}

std::cout << "The reverse is ";//making reverse

for (int i = count; i >= 0; i--)

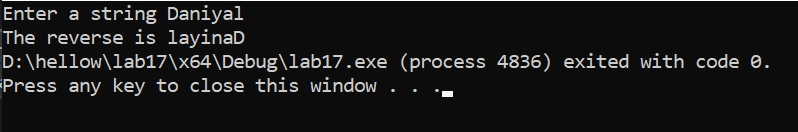
{

std::cout << str[i];

}

}

# Output



# Task 2

#include<iostream>

#include<string>

int main()

{

std::string str1 ,str2;//taking inputs

std::cout << "enter 1st string ";

std::getline(std::cin, str1);

std::cout << "enter 2nd string ";

std::getline(std::cin, str2);

int length= str1.length(); //using all functions

std::cout << "the length of string is " <<length<<"\n";

std::cout <<str1.substr() << "\n";;

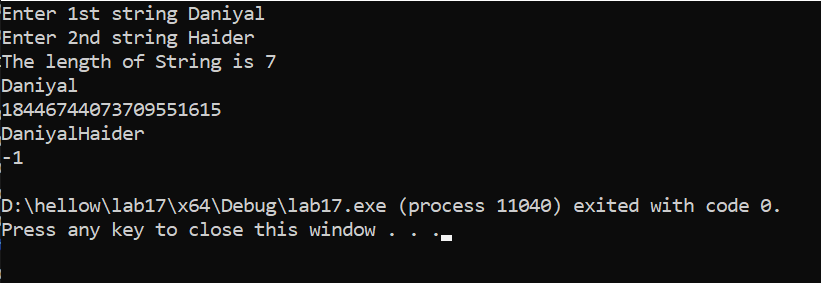
std::cout << str1.find("ali") << "\n";;

std::cout <<str1.append(str2) << "\n";;

std::cout << str1.compare(str2) << "\n";;

}

# Output



# Task 3

#include<iostream>

#include<string>

std::string position(std::string str, int a, int b);//declare

int main()

{

int startindex , endindex;

std::string str; //taking inputs

std::cout << "Enter a String ";

std::getline(std::cin, str);

std::cout << "Enter Starting index ";

std::cin >> startindex;

std::cout << "Enter Ending index ";

std::cin >> endindex;

str = position(str, startindex, endindex);//calling function

std::cout << "After Function " << str;

}

std::string position(std::string str, int a, int b) //defining

{

int count=0;

for (int i = 0; str[i] != '\0'; i++)

{

count++;

}

if (a < count && b < count)

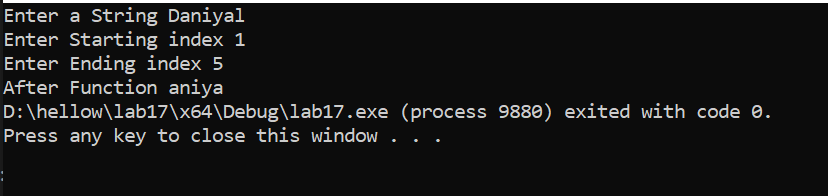
return str.substr(a, b);

else

return str = "null";

}

# Output



# Task 4

#include<iostream>

#include<string>

int main()

{

std::string str;

std::cout << "Enter your Name = ";

std::getline(std::cin, str);

int count = 0;

for (int i = 0; str[i] != '\0'; i++) //counting

{

count++;

}

for (int i = 0; str[i] != '\0'; i++)

{

if (count <= 3 || count >= 30)

{

std::cout << "Error ! Size Excede"; // checking

break;

}

else if (((str[i] >= 48 && str[i] <= 57) ||

(str[i] >= 32 && str[i] <= 47) ||

(str[i] >= 58 && str[i] <= 64) ||

(str[i] >= 91 && str[i] <= 96) ||

(str[i] >= 123 && str[i] <= 126)))

{

std::cout << "Error! Number or special character is entered ";

break;

}

else if (!(str[0] >= 65 && str[0] <= 90)) // for capitals

{

std::cout << "Error ! First letter Not capital ";

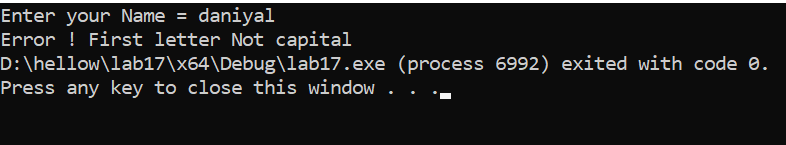
break;

}

}

}

# Output



# Task 5

#include<iostream>

#include<fstream>

int main()

{

std::string name="Daniyal";

int cgpa = 4, rollno=3029;

std::ofstream out; // handler

out.open("firstfile.txt"); //opening file

out << "Name : "; //writing name

out << name << "\n";

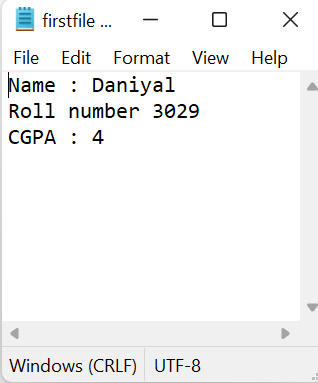
out << "Roll number " << rollno <<"\n";

out << "CGPA : " << cgpa << "\n";

out.close(); //closing file

}

# Output



# Task 6

#include<iostream>

#include<fstream>

int main()

{

std::ifstream in; // handler

in.open("firstfile.txt"); //opening file

char ch;

while (!in.eof()) //taking data

{

in.get(ch);

std::cout << ch;

}

in.close(); //closing file

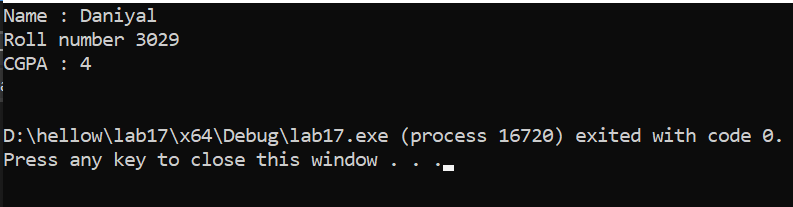
std::ofstream out("firstfile.txt", std::ios::trunc); //clearing file

out.close();

return 0;

}

# Output



# Task 7

#include<iostream>

#include<fstream>

int main()

{

int num1, num2, sum = 0;

std::cout << "Enter Two Numbers ";

std::cin >> num1 >> num2;

sum = num1 + num2;

std::ofstream out; // handler

out.open("sum.txt"); //opening file

out << "Number 1 " << num1 << "\n";

out << "Number 2 " << num2 << "\n";

out << "Sum is = " << sum << "\n";

out.close(); //closing file

std::ifstream in; // handler

in.open("sum.txt"); //opening file

char ch;

while (!in.eof()) //taking data

{

in.get(ch);

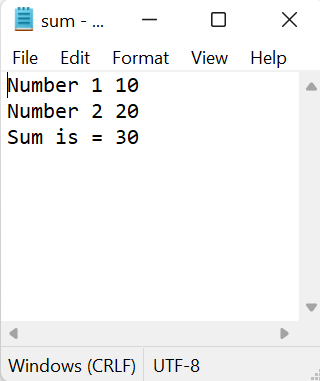
std::cout << ch;

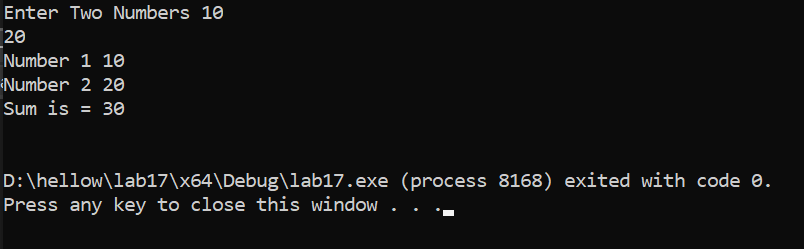
}

in.close(); //closing file

}

# Output





# Task 8

#include<iostream>

#include<fstream>

int main()

{

std::cout << "Enter a Number to print the Table"; //taking num

int num;

std::cin >> num;

std::ofstream out; //sving table in file

out.open("table.txt");

for (int i = 1; i <= 10; i++)

{

out << num << " \* " << i << " = " << num \* i << "\n";

}

out.close();

std::ifstream in;

in.open("table.txt");

char ch;

while (!in.eof()) //showing it on console

{

in.get(ch);

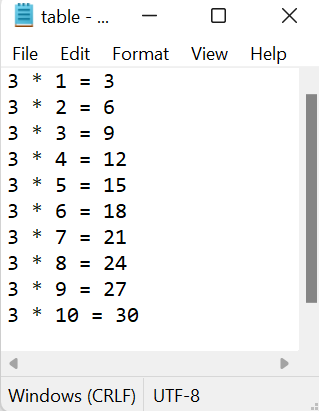
std::cout << ch;

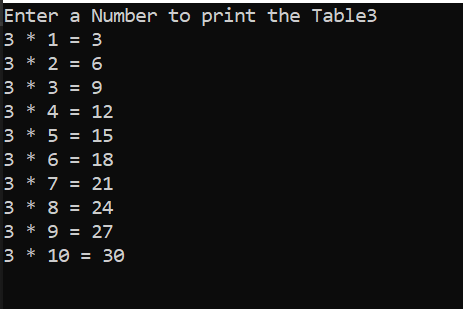
}

in.close();

}

# Output





# Task 9

#include<iostream>

#include<fstream>

#include<ctime>

int main()

{

srand(time(0)); //genrating 100 random numbers

int arr[100];

for (int i = 0; i < 100; i++)

{

arr[i] = ( rand() % 1000 ) + 1;

}

for (int i = 0; i <= 99; i++) //sorting them

{

for (int j = 0; j <= 99; j++)

{

if (arr[j] > arr[i])

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

for (int i = 0; i < 100; i++) //checking on console

{

std::cout << arr[i] << " ";

}

std::ofstream out; //saving them on file

out.open("sortednumbers.txt");

out << "Numbers in sorted Order \n" ;

for (int i = 0; i < 100; i++)

{

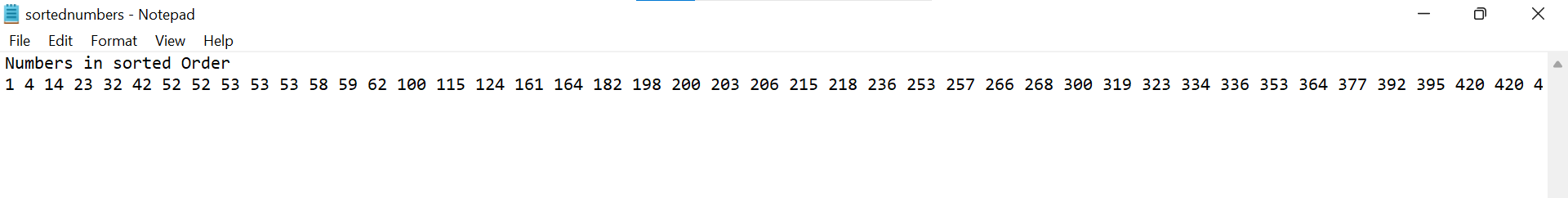
out << arr[i] << " ";

}

out.close();

}

# Output



# Task 10

#include <iostream>

#include <string>

float calculatePrice(float price, int quantity); //declaring + overloading

float calculatePrice(float price, int quantity, float disc);

float calculatePrice(float price, int quantity, int dilvery);

float calculatePrice(float price, int quantity, float disc, float dilevry);

int main()

{

float price = 10.0;

int quantity = 5;

float total = calculatePrice(price, quantity); //calling

std:: cout << "Final price is = " << total <<std:: endl;

price = 10.0;

quantity = 5;

float disc = 20.0;

total = calculatePrice(price, quantity, disc); //diffrent senerio

std::cout << "Final price After Discount = " << total << std::endl;

price = 10.0;

quantity = 5;

float dilevry = 5.0;

total = calculatePrice(price, quantity, dilevry);//another senerio

std::cout << "Final price After Dilevry Charges = " << total << std::endl;

price = 10.0;

quantity = 5;

disc = 20.0;

dilevry = 5.0;

total = calculatePrice(price, quantity, disc, dilevry); //another senerio which shows overloading

std::cout << "Final price For Special offers = " << total << std::endl;

return 0;

}

float calculatePrice(float price, int quantity) //defininng functions

{

return price \* quantity;

}

float calculatePrice(float price, int quantity, float disc)

{

disc = price \* (1 - disc / 100);

return disc \* quantity;

}

float calculatePrice(float price, int quantity, int delivery)

{

return (price \* quantity) + delivery;

}

float calculatePrice(float price, int quantity, float disc, float delivery)

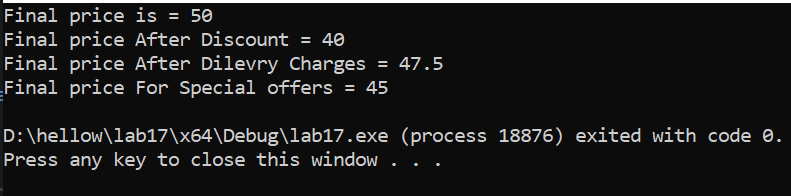
{

disc = price \* (1 - disc / 100);

return (disc \* quantity) + delivery;

}

# Output



# Task 11

#include <iostream>

#include <cstdlib>

#include <ctime>

void addQuestions(int& num1, int& num2, int& answer); //declaring prototype

void addQuestions(int& num1, int& num2, int& num3, int& answer);

void subQuestions(int& num1, int& num2, int& answer);

void multiplyQuestions(int& num1, int& num2, int& answer);

void divideQuestions(int& num1, int& num2, int& answer);

int main()

{

srand(time(0));

for (int i = 0; i < 5; ++i) //genrating quiz

{

int n1, n2, n3, ans , userAnswer;

int questionType = rand() % 5;

switch (questionType)

{

case 0:

addQuestions(n1, n2, ans);

std::cout << "Question " << i + 1 << ": " << n1 << " + " << n2 << " = ?" << std::endl;

std::cin >> userAnswer;

if (userAnswer == ans)

{

std::cout << "correct!!\n";

}

else

{

std::cout << "Incorrect!!\n";

}

break;

case 1:

subQuestions(n1, n2, ans);

std::cout << "Question " << i + 1 << ": " << n1 << " - " << n2 << " = ?" << std::endl;

std::cin >> userAnswer;

if (userAnswer == ans)

{

std::cout << "correct!!\n";

}

else

{

std::cout << "Incorrect!!\n";

}

break;

case 2:

multiplyQuestions(n1, n2, ans);

std::cout << "Question " << i + 1 << ": " << n1 << " \* " << n2 << " = ?" << std::endl;

std::cin >> userAnswer;

if (userAnswer == ans)

{

std::cout << "correct!!\n";

}

else

{

std::cout << "Incorrect!!\n";

}

break;

case 3:

divideQuestions(n1, n2, ans);

std::cout << "Question " << i + 1 << ": " << n1 << " / " << n2 << " = ?" << std::endl;

std::cin >> userAnswer;

if (userAnswer == ans)

{

std::cout << "correct!!\n";

}

else

{

std::cout << "Incorrect!!\n";

}

break;

case 4:

addQuestions(n1, n2, n3, ans);

std::cout << "Question " << i + 1 << ": " << n1 << " + " << n2 << " + " << n3 << " = ?" << std::endl;

std::cin >> userAnswer;

if (userAnswer == ans)

{

std::cout << "correct!!\n";

}

else

{

std::cout << "Incorrect!!\n";

}

break;

}

}

return 0;

}

//defining all

void addQuestions(int& num1, int& num2, int& answer)

{

num1 = rand() % 100;

num2 = rand() % 100;

answer = num1 + num2;

}

void addQuestions(int& num1, int& num2, int& num3, int& answer) {

num1 = rand() % 100;

num2 = rand() % 100;

num3 = rand() % 100;

answer = num1 + num2 + num3;

}

void subQuestions(int& num1, int& num2, int& answer) {

num1 = rand() % 100;

num2 = rand() % num1;

answer = num1 - num2;

}

void multiplyQuestions(int& num1, int& num2, int& answer) {

num1 = rand() % 20;

num2 = rand() % 20;

answer = num1 \* num2;

}

void divideQuestions(int& num1, int& num2, int& answer) {

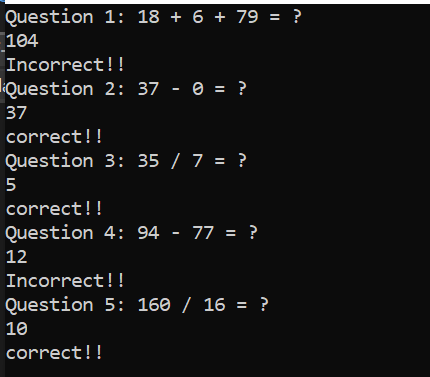
num2 = rand() % 19 + 1;

answer = rand() % 20;

num1 = num2 \* answer;

}

# Output



# Task 12

#include <iostream>

const int rows = 10;

const int cols = 10;

void read(int matrix[rows][cols], int rows, int cols); //declaring functions

void print(const int matrix[rows][cols], int rows, int cols);

void addMatrices(const int matrix1[rows][cols], const int matrix2[rows][cols], int result[rows][cols], int rows, int cols);

void multiplyMatrices(const int matrix1[rows][cols], const int matrix2[rows][cols], int result[rows][cols], int rows1, int cols1, int rows2, int cols2);

void transposeMatrix(const int matrix[rows][cols], int result[cols][rows], int rows, int cols);

int main()

{

int matrix1[rows][cols];

int matrix2[rows][cols]; //declaring arrays

int resultMatrix[rows][cols];

int transposeResult[cols][rows];

int rows1, cols1, rows2, cols2;

std::cout << "Enter the number of rows and columns for the first matrix: "; //taking inputs

std::cin >> rows1 >> cols1;

std::cout << "Enter elements for the first matrix:\n";

read(matrix1, rows1, cols1);

std::cout << "\nEnter the number of rows and columns for the second matrix: ";

std::cin >> rows2 >> cols2;

std::cout << "Enter elements for the second matrix:\n";

read(matrix2, rows2, cols2);

std::cout << "\nMatrix 1:\n";

print(matrix1, rows1, cols1);

std::cout << "\nMatrix 2:\n";

print(matrix2, rows2, cols2);

addMatrices(matrix1, matrix2, resultMatrix, rows1, cols1); //calling Functions

std::cout << "\nMatrix Addition Result:\n";

print(resultMatrix, rows1, cols1);

multiplyMatrices(matrix1, matrix2, resultMatrix, rows1, cols1, rows2, cols2);

std::cout << "\nMatrix Multiplication Result:\n";

print(resultMatrix, rows1, cols2);

transposeMatrix(matrix1, transposeResult, rows1, cols1);

std::cout << "\nTranspose of Matrix 1:\n";

print(transposeResult, cols1, rows1);

return 0;

}

void read(int matrix[rows][cols], int rows, int cols) //defining All Functions

{

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

{

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

{

std::cin >> matrix[i][j];

}

}

}

void print(const int matrix[rows][cols], int rows, int cols)

{

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

{

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

{

std::cout << matrix[i][j] << " ";

}

std::cout << '\n';

}

}

void addMatrices(const int matrix1[rows][cols], const int matrix2[rows][cols], int result[rows][cols], int rows, int cols)

{

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

{

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

{

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

}

void multiplyMatrices(const int matrix1[rows][cols], const int matrix2[rows][cols], int result[rows][cols], int rows1, int cols1, int rows2, int cols2)

{

for (int i = 0; i < rows1; ++i)

{

for (int j = 0; j < cols2; ++j)

{

result[i][j] = 0;

for (int k = 0; k < cols1; ++k)

{

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

}

void transposeMatrix(const int matrix[rows][cols], int result[cols][rows], int rows, int cols)

{

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

{

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

{

result[j][i] = matrix[i][j];

}

}

}

# Output

