Q1)

#include <iostream>

using namespace std;

template <class T>

T Large(T n1, T n2){

return (n1 > n2) ? n1 : n2;

}

int main(){

int i1, i2;

float f1, f2;

char c1, c2;

cout << "Enter two integers:\n";

cin >> i1 >> i2;

cout << Large(i1, i2) <<" is larger." << endl;

cout << "\nEnter two floating-point numbers:\n";

cin >> f1 >> f2;

cout << Large(f1, f2) <<" is larger." << endl;

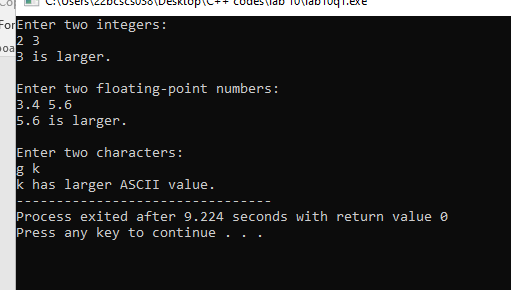
cout << "\nEnter two characters:\n";

cin >> c1 >> c2;

cout << Large(c1, c2) << " has larger ASCII value.";

return 0;

}



Q2)

#include <iostream>

using namespace std;

template<typename T>

double average(T arr[], int size) {

double sum = 0;

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

sum += static\_cast<double>(arr[i]);

}

return sum / size;

}

int main() {

const int SIZE = 5;

int intArr[SIZE];

double doubleArr[SIZE];

cout << "Enter " << SIZE << " integer numbers: ";

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

cin >> intArr[i];

}

cout << "Average (int): " << average(intArr, SIZE) << endl;

cout << "Enter " << SIZE << " double numbers: ";

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

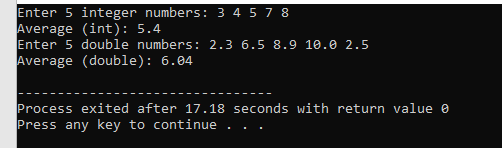
cin >> doubleArr[i];

}

cout << "Average (double): " << average(doubleArr, SIZE) << endl;

return 0;

}



Q3)

#include <iostream>

#include <stdexcept>

using namespace std;

double divideNumbers(int dividend, int divisor) {

if (divisor == 0) {

throw runtime\_error("Division by zero error");

}

return static\_cast<double>(dividend) / divisor;

}

int main() {

int dividend, divisor;

cout << "Enter dividend: ";

cin >> dividend;

cout << "Enter divisor: ";

cin >> divisor;

try {

double result = divideNumbers(dividend, divisor);

cout << "Result: " << result << endl;

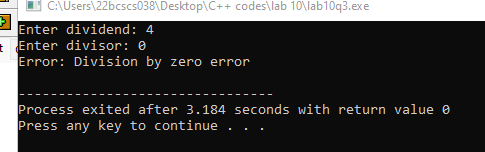
} catch (const exception& e) {

cout << "Error: " << e.what() << endl;

}

return 0;

}



Q4)

#include <iostream>

#include <stdexcept>

using namespace std;

int divide(int a, int b) {

if (b == 0) {

throw invalid\_argument("Division by zero is not allowed");

}

return a / b;

}

int main() {

int dividend, divisor;

cout << "Enter the dividend: ";

cin >> dividend;

cout << "Enter the divisor: ";

cin >> divisor;

try {

int result = divide(dividend, divisor);

cout << "Result: " << result << endl;

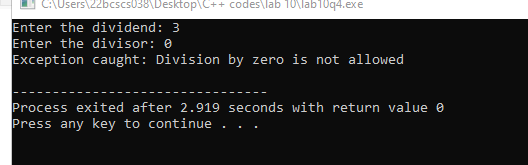
} catch (const invalid\_argument& e) {

cout << "Exception caught: " << e.what() << endl;

}

return 0;

}



Q5)

#include <iostream>

#include <stdexcept>

using namespace std;

class Student {

private:

string name;

int age;

public:

void setAge(int newAge);

void setName(const string& newName);

void displayInfo();

};

void Student::setAge(int newAge) {

if (newAge < 18 || newAge > 30) {

throw out\_of\_range("Invalid age. Age must be between 18 and 30.");

}

age = newAge;

}

void Student::setName(const string& newName) {

name = newName;

}

void Student::displayInfo() {

cout << "Name: " << name << endl;

cout << "Age: " << age << endl;

}

int main() {

Student student;

string name;

int age;

cout << "Enter student name: ";

getline(cin, name);

cout << "Enter student age: ";

cin >> age;

cin.ignore();

try {

student.setName(name);

student.setAge(age);

student.displayInfo();

} catch (const out\_of\_range& e) {

cout << "Exception caught: " << e.what() << endl;

return 1;

}

return 0;

}

