TASK 1

#include    <iostream>

#include    <string>

template <typename T>

T maxValue (T a, T b){

    return (a>b) ? a:b;

}

int main(){

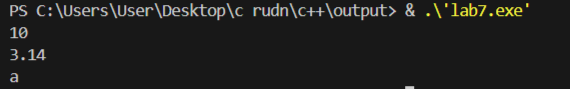
    std::cout<<maxValue(5,10)<<std::endl;

    std::cout<<maxValue(3.14, 2.85)<<std::endl;

    std::cout<<maxValue("a","o")<<std::endl;

    return 0;

}



TASK 2

#include <iostream>

#include <type\_traits>

template <typename T, typename U>

bool isEqual(T a, U b){

    static\_assert(std::is\_same<T, U>::value, "Error:    Types   must    be  the same!");

    return a==b;

}

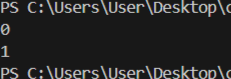
int main(){

    std::cout<<isEqual(6,5)<<std::endl;

    std::cout<<isEqual(5,5)<<std::endl;

    return 0;

}



TASK 3

#include    <iostream>

#include    <typeinfo>

template <typename T>

void printType(T value) {

                std::cout<<"Type:   "<< typeid(T).name()<<std::endl;

}

template <>

void printType<const char\*>(const char\* value) {

                std::cout<<"Type:   C-style string"<<std::endl;

}

int main() {

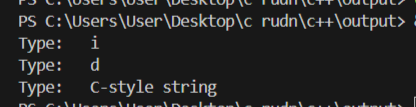
    printType(42);

    printType(3.14);

    printType("Hello, world!");

    return 0;

}



TASK 4

#include    <iostream>

template <typename T, typename U>

auto multiply(T a, U b) -> decltype(a \* b) { //тип определится автоматически на этапе компиляции

    return a \* b;

}

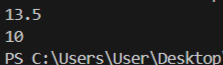
int main() {

    std::cout<< multiply(3, 4.5)<<std::endl;            //  13.5    (double)

    std::cout<< multiply(2.5, 4)<<std::endl;            //  10.0    (double)

    return 0;

}



TASK 5

#include    <iostream>

template <typename T>

T   square(T x) {

    return x \* x;

}

template <>

int square<int>(int x) {

    std::cout<< "Squaring an int: ";

    return x \* x;

}

int main() {

    std::cout<<square(5.5)<<std::endl;      //  30.25

    std::cout<<square(5)<<std::endl;        //  "Squaring   an  int:    25"

    return 0;

}

