Student Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student Number \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Closed Book. Write your answers to all questions on this sheet. Marks are in parentheses.

1. (3) What is the output of this program

#include <iostream>

#include <thread>

#include <string>

using namespace std;

thread\_local int x{ 10 }; //thread\_local will make x available for each thread initialized with x = 10

string counting[2] = { "one", "two" };

void task(int ss, int id) {

x++;

}

int main() {

thread t1(task, 0, x-- );

t1.join();

thread t2(task, 1, x-- );

t2.join();

for (auto c : counting)

cout << c << endl;

cout << x;

}

1. (6) Find 3 errors in the program

#include <iostream>

#include <string>

using namespace std;

struct Chair

{

string name;

static size\_t cnt;

Chair(string n) : name(n) { ++cnt; }

};

size\_t Chair::cnt = 0; //solution

struct Rocking : public Chair

{

Rocking(string n = "")

: Chair(n) {}

};

int main()

{

int a[][2] = { {1, 2}, {3, 4, 5} }; // int a[][2] = { {1,2},{3,4}}; solution

Rocking nc("Rocking\_Chair");

Rocking\* c = new Rocking("rchair");

cout << " Chair name: " << c.name << endl; // c -> name; solution

cout << Chair.cnt << endl; // Chair::cnt; solution

delete a; // delete not needed

}

1. (5) The following function uses raw pointer; write the equivalent function that uses unique smart pointer (assume the project contains a class Foo, that has a public member function named doWork()). Include any necessary header in your solution.



|  |
| --- |
| void executeRaw()  {  Foo\* theFoo = new Foo("Hello");  theFoo->doWork();  delete[] theFoo;  } |
| void executeSmart() |

{

std::unique\_ptr <Foo> sp(new Foo("Hello"));

sp->doWork();

}

1. (10) Design and implement a function named printStats(…) that receives a reference to an STL vector of numbers, and—using the algorithms library and lambda expressions—prints the answer to the following questions:

Important: Your implementation should not include any manual loops—use the algorithms library to process the collection. For reference, see the functions on the last page.

* How many numbers bigger than 10 are in the vector?
* What is the average of all numbers in the vector?

#include <numeric>

#include <algorithm>

#include <iostream>

#include<vector>

int main() {

std::vector<int> a = {1, 12, 4, 5, 8, 9, 12, 13, 16, 18, 12};

int n = std::count\_if(a.begin(), a.end(), [=](int x) {return x > 10; });

std::cout << "Number greater than 10 are: " << n << std::endl;

double avg = 0.0;

double s = std::accumulate(a.begin(), a.end(), 0.0);

std::cout << "The total is: " << s << std::endl;

avg = s / a.size();

std::cout << "The average is: " << avg <<std::endl;

}