X

Week 11: Assignment 11

The due date for submitting this assignment has passed.

Due on 2025-04-09, 23:59 IST.

As per our records you have not submitted this assignment.

1) To write a C++ program that uses std::thread to execute two functions concurrently: ***1 point***

- printHello() prints ”Hello” five times.
- printWorld() prints ”World” five times.

What is the correct way to join threads after starting them?

Use thread1.start() and thread1.join().

Use thread1.join() and thread2.join().

Use thread1.detach() and thread2.detach().

Threads cannot be joined in C++.

No, the answer is incorrect.
Score: 0

Accepted Answers:
Use thread1.join() and thread2.join().

2) Which of the following is true about the Runnable interface in Java? ***1 point***

It has a method called run() that must be overridden.

t can only be used with the Thread class.

It supports multiple inheritance.

It cannot be implemented in a lambda expression.

No, the answer is incorrect.
Score: 0

Accepted Answers:
It has a method called run() that must be overridden.

3) To write a Java program that creates two threads to: ***1 point***

- Print numbers from 1 to 5 in one thread.
- Print the squares of numbers from 1 to 5 in another thread.

Which of the following correctly starts both threads?

new Thread(thread1).run(); new Thread(thread2).run();

thread1.start(); thread2.start();

- new Thread(thread1).start(); new Thread(thread2).start();
- thread1.run(); thread2.run();

No, the answer is incorrect.
Score: 0

Accepted Answers:
new Thread(thread1).start(); new Thread(thread2).start();

4) To write a Python program that creates a TCP server to: ***1 point***

- Accept connections from clients.
- Receive a message from the client and print it.
- Send an acknowledgment back to the client.

Which of the following is the correct method to bind the server to a port?

- server.bind((’localhost’, 8080))
- server.listen(8080)
- server.start((’localhost’, 8080))
- server.connect((’localhost’, 8080))

No, the answer is incorrect.
Score: 0

Accepted Answers:
server.bind((’localhost’, 8080))

5) To write a Java Swing program that creates a window with: ***1 point***

- A JButton labeled ”Click Me”.
- An event listener that displays ”Button Clicked” in the console when the button is clicked.

Which of the following methods is used to add an event listener to the button?

- button.addActionListener()
- button.addListener()
- button.onClick()
- button.setActionListener()

No, the answer is incorrect.
Score: 0

Accepted Answers:
button.addActionListener()

6) To write a program to: ***1 point***

- Create a multithreaded TCP server that handles multiple clients simultaneously.
- Use threads to process client requests independently.

Which of the following is essential for the server to handle multiple clients?

- Use thread.join() for each client connection.
- Use a separate thread for each client connection.
- Use a single thread for all client connections.
- Use the poll() function to manage threads.

No, the answer is incorrect.
Score: 0

Accepted Answers:
Use a separate thread for each client connection.

7) Consider the following code that implements an Observer Pattern for monitoring earthquake magnitudes. What will be the output of the program? ***1 point***

```
#include <iostream>
#include <vector>
using namespace std;
class Observer {
public:
    virtual void update(double magnitude) = 0;
};
class EarthquakeMonitor : public Observer {
public:
    void update(double magnitude) override {
        if (magnitude > 5.0)
            cout << "Alert: Significant earthquake of magnitude "
                 << magnitude << " detected!" << endl;
    }
};
class Subject {
    vector<Observer*> observers;
    double magnitude;
public:
    void attach(Observer* obs) { observers.push_back(obs); }
    void notify() {
        for (Observer* obs : observers) obs->update(magnitude);
    }
    void setMagnitude(double mag) {
        magnitude = mag;
        notify();
    }
};
int main() {
    Subject earthquakeData;
    EarthquakeMonitor monitor;
    earthquakeData.attach(&monitor);
    earthquakeData.setMagnitude(4.2);
    earthquakeData.setMagnitude(5.8);
```

```
    earthquakeData.setMagnitude(5.8);
    return 0;
}
```

- Alert: Significant earthquake of magnitude 4.2 detected!
- Alert: Significant earthquake of magnitude 5.8 detected!
- Alert: Significant earthquake of magnitude 5.8 detected!
- No output
- Runtime error: Null reference to observer

No, the answer is incorrect.
Score: 0

Accepted Answers:
Alert: Significant earthquake of magnitude 5.8 detected!

8) Consider the following Java code snippet: ***1 point***

```
class SharedResource {
    synchronized void display(String message) {
        for (int i = 0; i < 3; i++) {
            System.out.println(message + " " + i);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
class MyThread extends Thread {
    SharedResource resource;
    String message;
    MyThread(SharedResource resource, String message) {
        this.resource = resource;
        this.message = message;
    }
    public void run() {
        resource.display(message);
    }
}
public class Main {
    public static void main(String[] args) {
        SharedResource resource = new SharedResource();
        MyThread t1 = new MyThread(resource, "Thread-1");
        MyThread t2 = new MyThread(resource, "Thread-2");
        t1.start();
        t2.start();
    }
}
```

What will the output of the program?

- Both Thread-1 and Thread-2 will interleave their output, as synchronization only applies to individual iterations of the loop.
- Only Thread-1 will execute its complete task, followed by Thread-2.
- Both threads will execute concurrently without any synchronization effects.
- Both threads will execute their tasks in sequence, with Thread-1 completing first and Thread-2 starting only after Thread-1 finishes.

No, the answer is incorrect.
Score: 0

Accepted Answers:
Both threads will execute their tasks in sequence, with Thread-1 completing first and Thread-2 starting only after Thread-1 finishes.

9) Which scenario causes a thread to transition directly from the ”Running” state to the ”Terminated” state in Java? ***1 point***

- The join() method is invoked on the thread object, causing the thread to stop execution immediately.
- The thread completes its execution of the run() method without encountering exceptions or interruptions.
- The sleep() method is called inside the thread, and the thread’s sleep time expires.
- The thread enters a synchronized block and encounters contention for a lock.

No, the answer is incorrect.
Score: 0

Accepted Answers:
The thread completes its execution of the run() method without encountering exceptions or interruptions.

10) What does the following code snippet demonstrate? ***1 point***

```
#include <iostream>
#include <future>
#include <chrono>
int computeValue() {
    std::this_thread::sleep_for(std::chrono::seconds(2));
    return 42;
}
int main() {
    auto futureValue = std::async(std::launch::async, computeValue);
    std::cout << "Processing..." << std::endl;
    std::cout << "Value: " << futureValue.get() << std::endl;
    return 0;
}
```

- Parallel computation using a new thread for computeValue().
- Deferred execution of computeValue() until futureValue.get() is called.
- Immediate execution of computeValue() in the main thread.
- Compilation error due to incorrect usage of std::async.

No, the answer is incorrect.
Score: 0

Accepted Answers:
Parallel computation using a new thread for computeValue().