Computer Programming 2 Lab

2023/02/22 Andy Hung

Outline

- HW 1
- From C to C++
- WSL

HW1

Description

Last semester, in HW6, you helped Tom convert the time code, and he was very grateful. However, he now has a new problem.

When Tom is searching for courses, he adds courses that he is interested in to his trace list. However, he is having difficulty resolving conflicts when he has to choose the order in which to take the courses.

HW1

In the above situation, the courses `[redacted]` and `[redacted]` have 2 hours of overlap (78), which is considered a **conflict**. Please help Tom count the number of conflicts he has.

HW1

Input

The first line contains a positive integer N, which represents the number of classes.

For each class, the first line provides the `` `session count` `` and `` `course ID` ``, and the second line provides the `` `weekday` ``, `` `start time` ``, and `` `end time` ``.

Constraints:

- N ≤ 10
- 1 ≤ session count ≤ 4
- |course ID| = 9
- 1 ≤ weekday ≤ 7
- 6 ≤ start time ≤ 21
- 7 ≤ end time ≤ 22

HW1

Output

Please print out how many conflict Tom has.

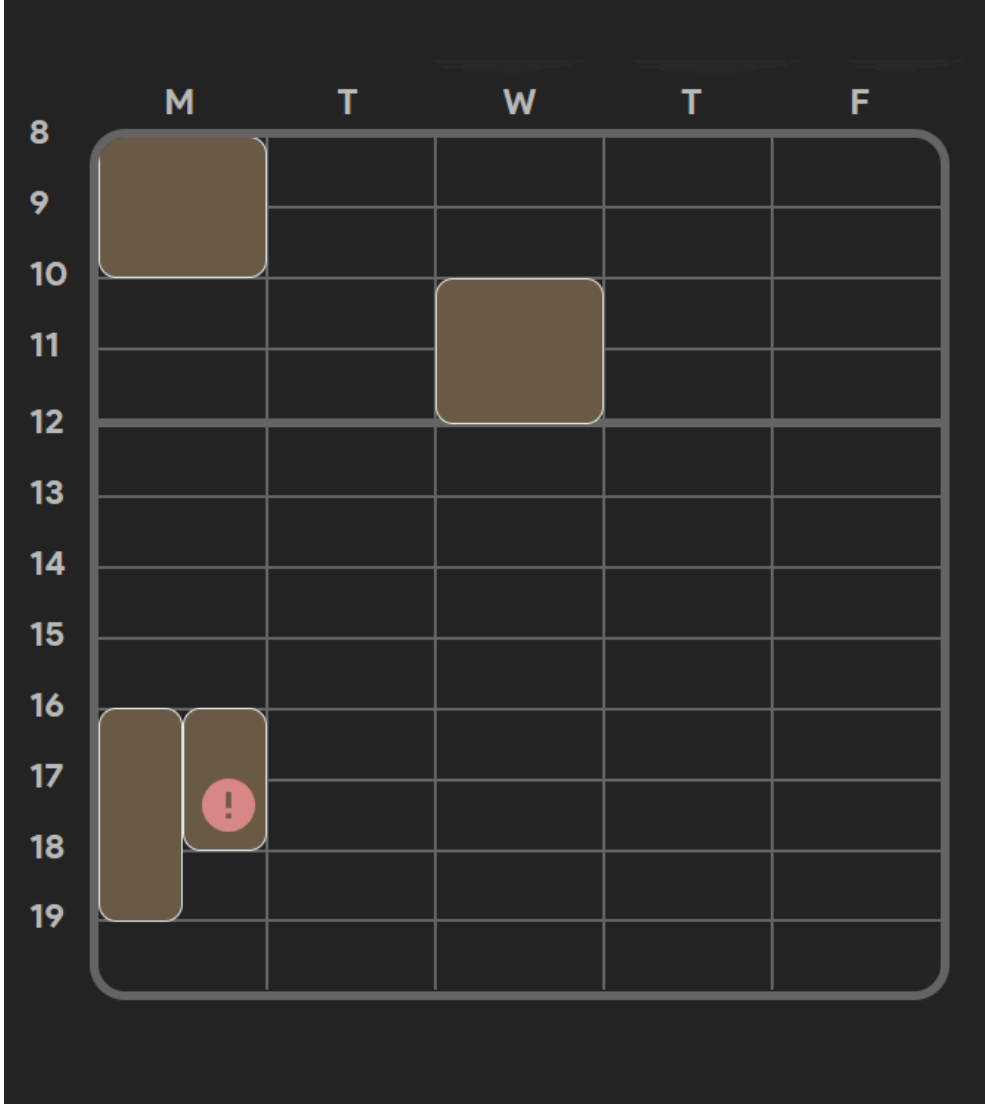
HW1

Input

```
4
1 002347001
1, 8, 10
1 002374001
3, 10, 12
1 042002001
1, 16, 19
1 044022001
1, 16, 18
```

Output

```
1
```



From C to C++: Hello World

```cpp
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!";
  return 0;
}
```

From C to C++: Hello World

```cpp
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!";
  return 0;
}
```

From C to C++: Hello World

```cpp
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!";
  return 0;
}
```

- A **namespace** is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it.

From C to C++: Hello World

```cpp
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!";
  return 0;
}
```

- A **namespace** is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it.
- All C++ standard library types and functions are declared in the std namespace or namespaces nested inside std.

From C to C++: Hello World

```cpp
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!";
  return 0;
}
```

- A **namespace** is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it.
- All C++ standard library types and functions are declared in the std namespace or namespaces nested inside std.
- No `.h` in `#include` most of the time.

From C to C++: I/O

- <iostream> includes commonly used `cin`, `cout`, `endl`.
- `cin` will read variables you defined automatically.
- `while(scanf( ... ) ≠ EOf)` ==== `while(cin >> a)`

```cpp
#include <iostream>
using namespace std;

int main() {
  int a;
  float b;
  string c;

  cin >> a >> b >> c; // input: 01 01.0 01.0
  cout << a << " " << b << " " << c << endl; // output: 1 1 01.0

  return 0;
}
```

From C to C++: I/O

- <iostream> includes commonly used `cin`, `cout`, `endl`.
- `cin` will read variables you defined automatically.
- `while(scanf( ... ) ≠ EOf)` ==== `while(cin >> a)`

```cpp
#include <iostream>
using namespace std;

int main() {
  int a;
  float b;
  string c;

  cin >> a >> b >> c; // input: 01 01.0 01.0
  cout << a << " " << b << " " << c << endl; // output: 1 1 01.0

  return 0;
}
```

From C to C++: I/O

- <iostream> includes commonly used `cin`, `cout`, `endl`.
- `cin` will read variables you defined automatically.
- `while(scanf( ... ) ≠ EOf)` ==== `while(cin >> a)`

```cpp
#include <iostream>
using namespace std;

int main() {
  int a;
  float b;
  string c;

  cin >> a >> b >> c; // input: 01 01.0 01.0
  cout << a << " " << b << " " << c << endl; // output: 1 1 01.0

  return 0;
}
```

From C to C++: String

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string str1 = "Hello";
  string str2 = "Hello";
  string str3 = "World";

  // String is comparable.
  cout << str1 == str2 << endl; // True
  cout << str1 == str3 << endl; // False

  // Strings can be concatenated.
  string str4 = str1 + str3;
  cout << str4 << endl;

  return 0;
}
```

From C to C++: String

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string str1 = "Hello";
  string str2 = "Hello";
  string str3 = "World";

  // String is comparable.
  cout << str1 == str2 << endl; // True
  cout << str1 == str3 << endl; // False

  // Strings can be concatenated.
  string str4 = str1 + str3;
  cout << str4 << endl;

  return 0;
}
```

From C to C++: String

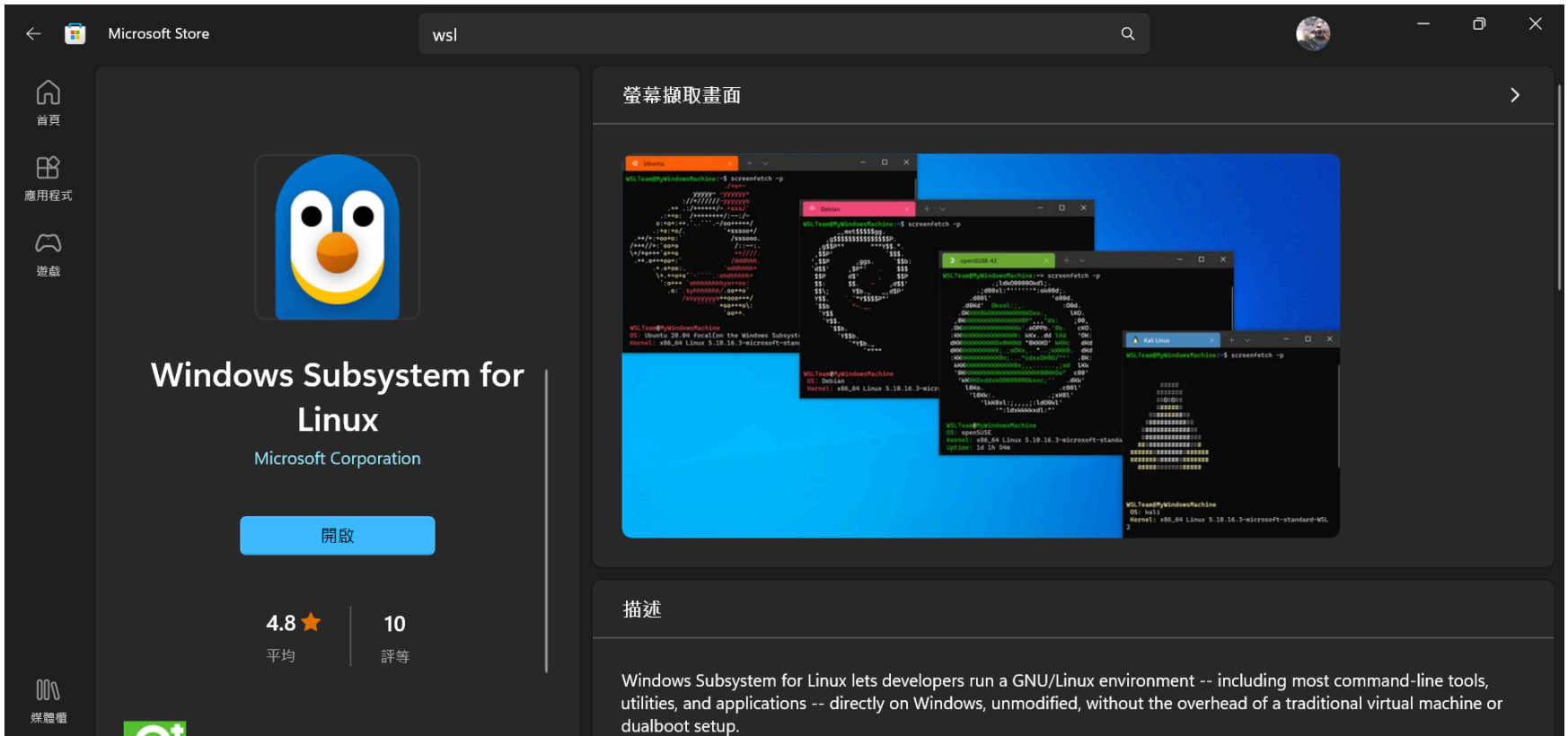
```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  string str1 = "Hello";
  string str2 = "Hello";
  string str3 = "World";

  // String is comparable.
  cout << str1 == str2 << endl; // True
  cout << str1 == str3 << endl; // False

  // Strings can be concatenated.
  string str4 = str1 + str3;
  cout << str4 << endl;

  return 0;
}
```

From C to C++: Vector

- C++'s new container, needs to include `` `<vector>` ``.
- Do not need to declare the size, but slower than array.
- Use `` `template` `` to declare vector type.

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
  vector<int> a;

  a.push_back(1); // a[0] = 1
  a.push_back(2); // a[1] = 2

  vector<vector<int>> 2d_array;

  return 0;
}
```

WSL


Microsoft Store
wsl
首頁
應用程式
遊戲
媒體櫃
Windows Subsystem for Linux
Microsoft Corporation
開啟
4.8
平均
10
評等
螢幕擷取畫面
描述
Windows Subsystem for Linux lets developers run a GNU/Linux environment -- including most command-line tools, utilities, and applications -- directly on Windows, unmodified, without the overhead of a traditional virtual machine or dualboot setup.

WSL

Installation

```
wsl --install
```

or download from microsoft store

```
wsl --install Ubuntu
```

WSL

WSL

- Can access Windows execution files

WSL

- Can access Windows execution files
- Can use Windows files

WSL

- Can access Windows execution files
- Can use Windows files
- Can setup server

WSL

- Can access Windows execution files
- Can use Windows files
- Can setup server
- Can have multiple sub-system

WSL

- Can access Windows execution files
- Can use Windows files
- Can setup server
- Can have multiple sub-system
- Win11 has gWSL

WSL

- Can access Windows execution files
- Can use Windows files
- Can setup server
- Can have multiple sub-system
- Win11 has gWSL
- Can use Nividia CUDA

WSL

- Can access Windows execution files
- Can use Windows files
- Can setup server
- Can have multiple sub-system
- Win11 has gWSL
- Can use Nividia CUDA
- Some commands may not work (WSL-kerel)

Thanks for listening
Any Questions?