**ECPI University**

**CIS\_126**

**Rafat Khandaker**

**08/25/18**

**Unit 5 Graded Assignment**

**Step 1**

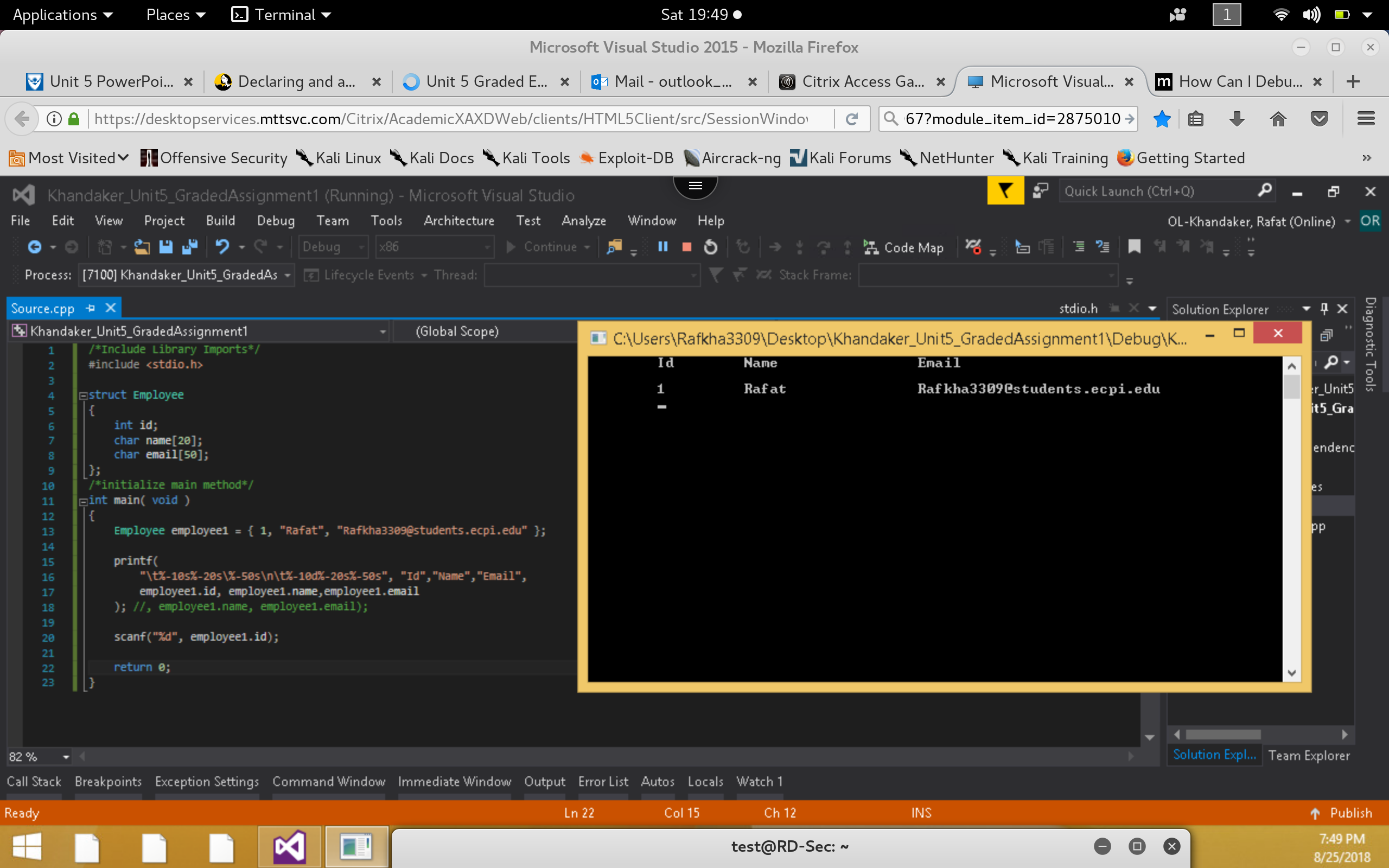
1. **Define a structure and identify its parts providing a syntax example.**

*A structure is a defined by naming a memory location to hold multiple variable types, similar to an array. A structure is initialized by defining a unique template name and defining it's elements inside a code block. The struct is terminated by a semi-colon, similar to initializing a variable. In real world logic, we can define a struct with variables that define an employee in a company & employee attributes, like: id, name & salary.*

1. **How is a structure initialized and accessed? Please provide a syntax example.**

***Example:***

*In this example I initiated a struct employee, where I am going to populate it’s value & print output from the struct values.*



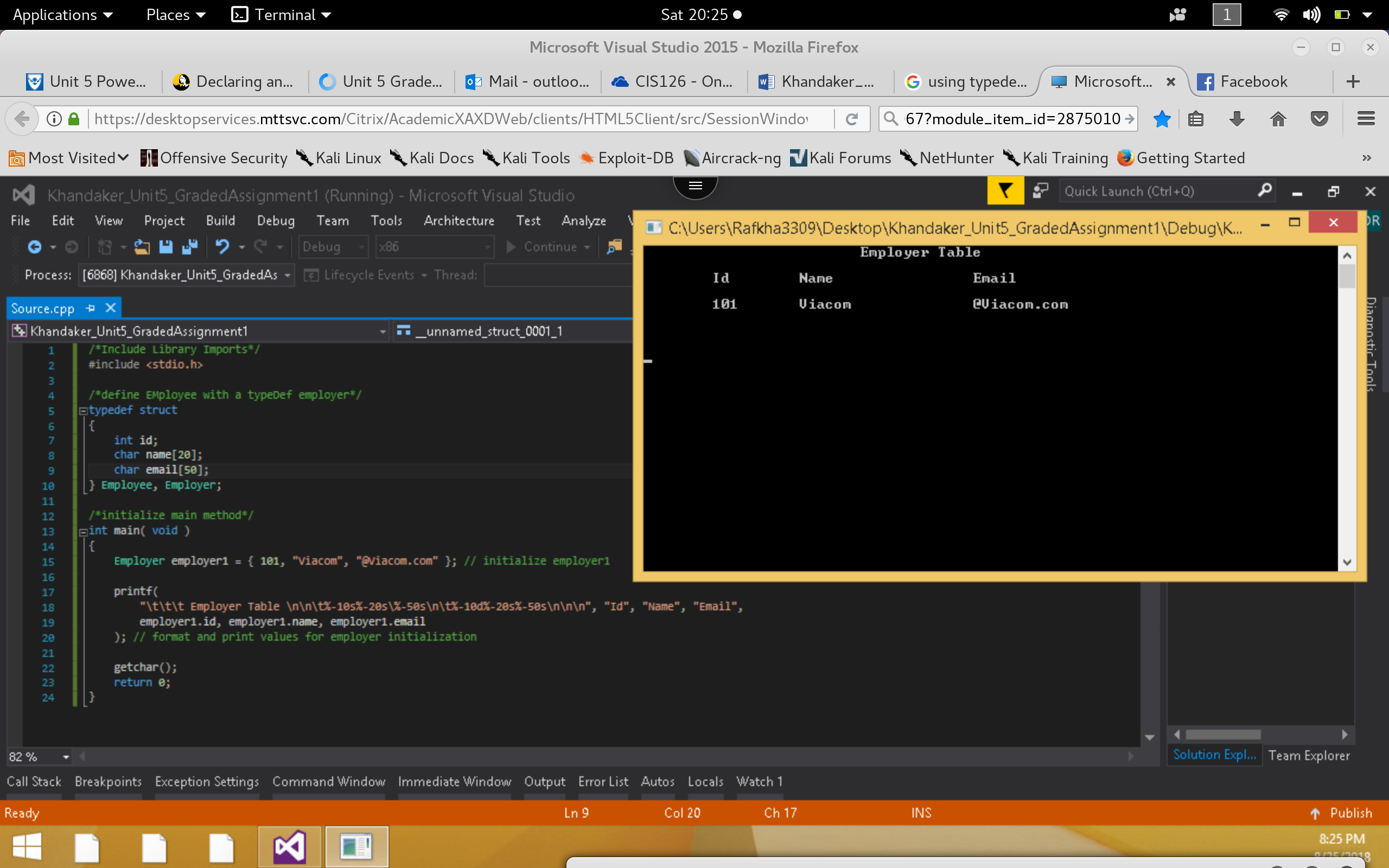
1. **Define the keyword *typedef* and explain how it can be used?**

*Using a struct typedef name can allow a programmer to define a struct as an alias, which can be Re-used in code to reference the struct through an alias. Programmers usually use typedef in order to create more readable code.*

***Example:***

*In this example, I used type Employer & employee to define a strut type containing*

*{ id, name, email }*



1. **What is a union and how can it be used?**

*A union is similar to a struct in initialization. It can be used to name a memory location, which can be replaced by multiple data types, defined. It can be thought of as a value that can be inherit multiple data-types.*

***Example:***

*In this example, we can see that the position in memory for the variable are being over-written. The actual value in memory.*

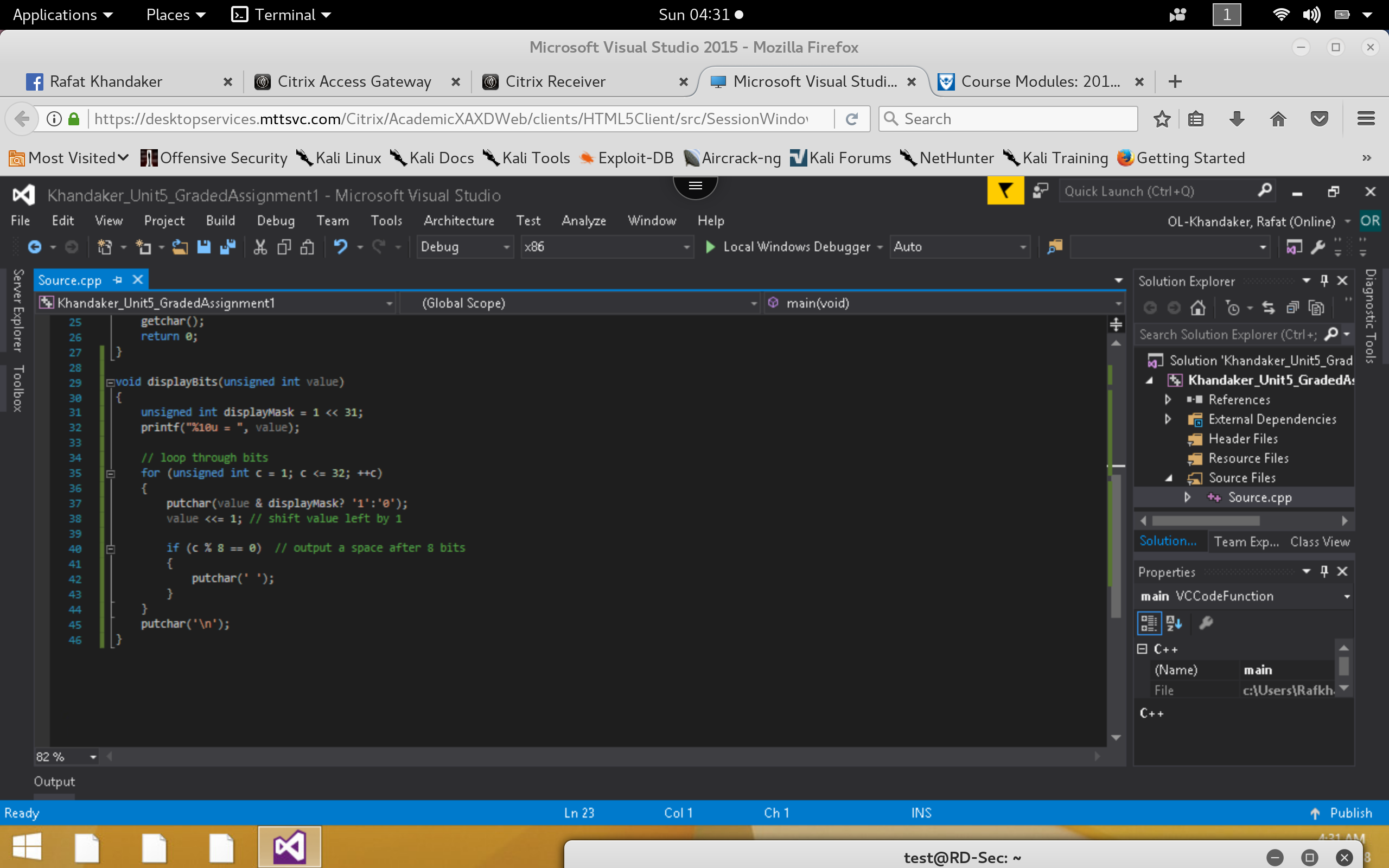
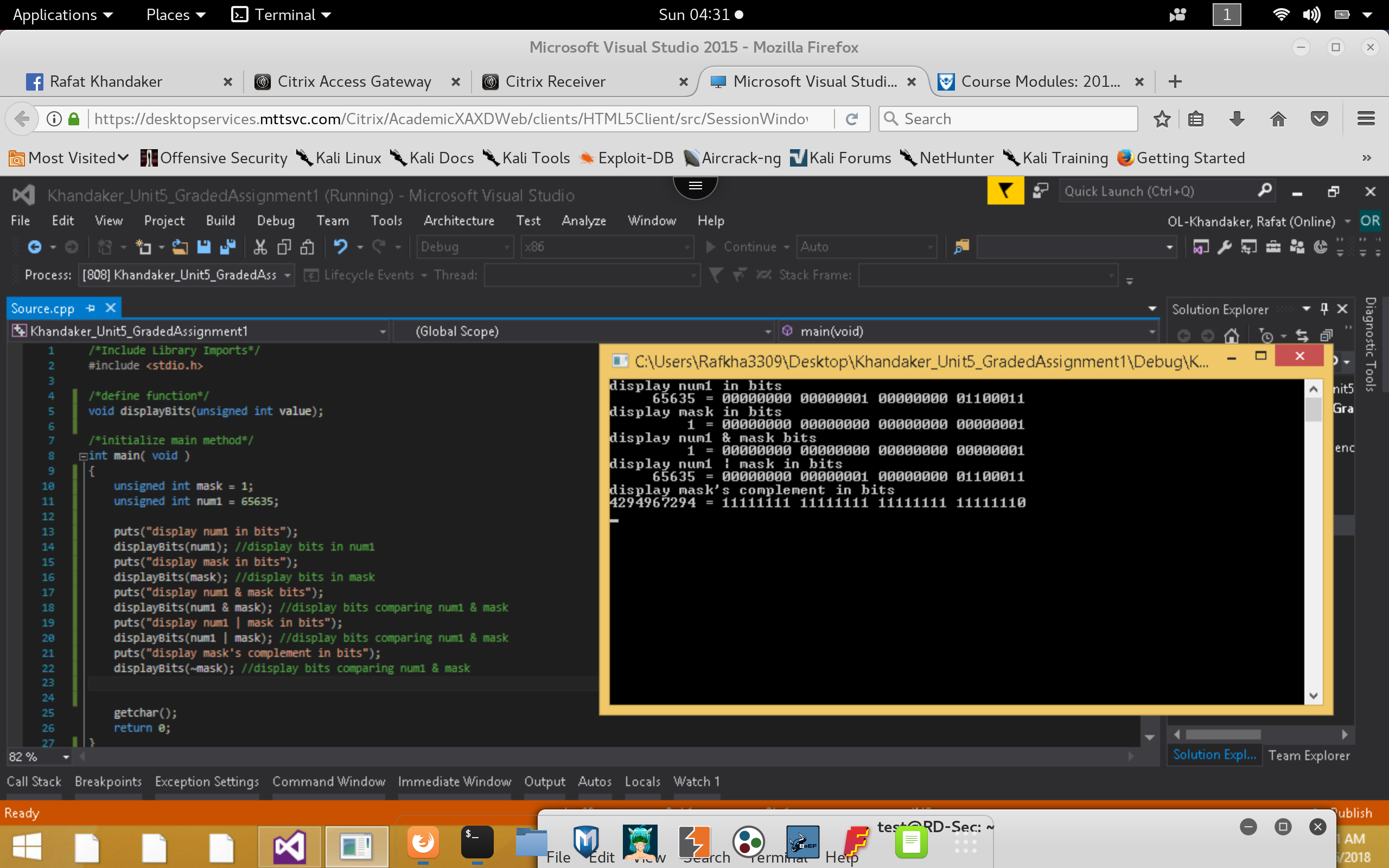


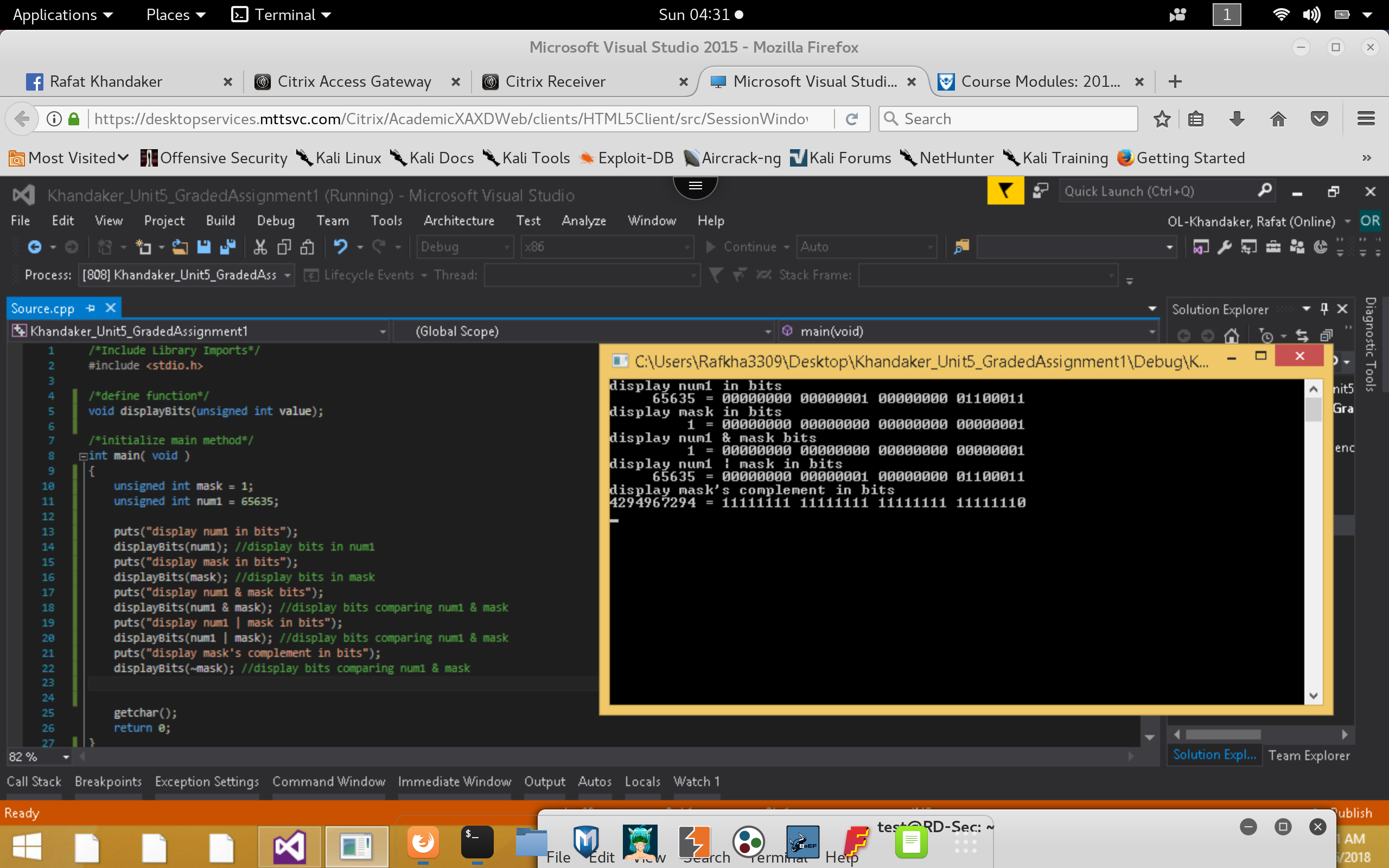
1. **Explain and provide an example of a bitwise operator.**

*Bit wise operators are the primitives of binary computation using Boolean logic. Boolean logic uses: And / Or / Xor / Complement, comparison to determine the output of a value. In a real world example, bit wise operator can be a subnet mask bits compared to an ip address, where both will be compared with the AND operator to determine the network address of a subnet. Programmatically, I provided an sample of such case:*

***Example:***

*In this example, I replicated an example, comparison of binary bit operation. Unsigned int is displayed into binary bits & compared with each bit using bitwise operator. The result is displayed on screen.*



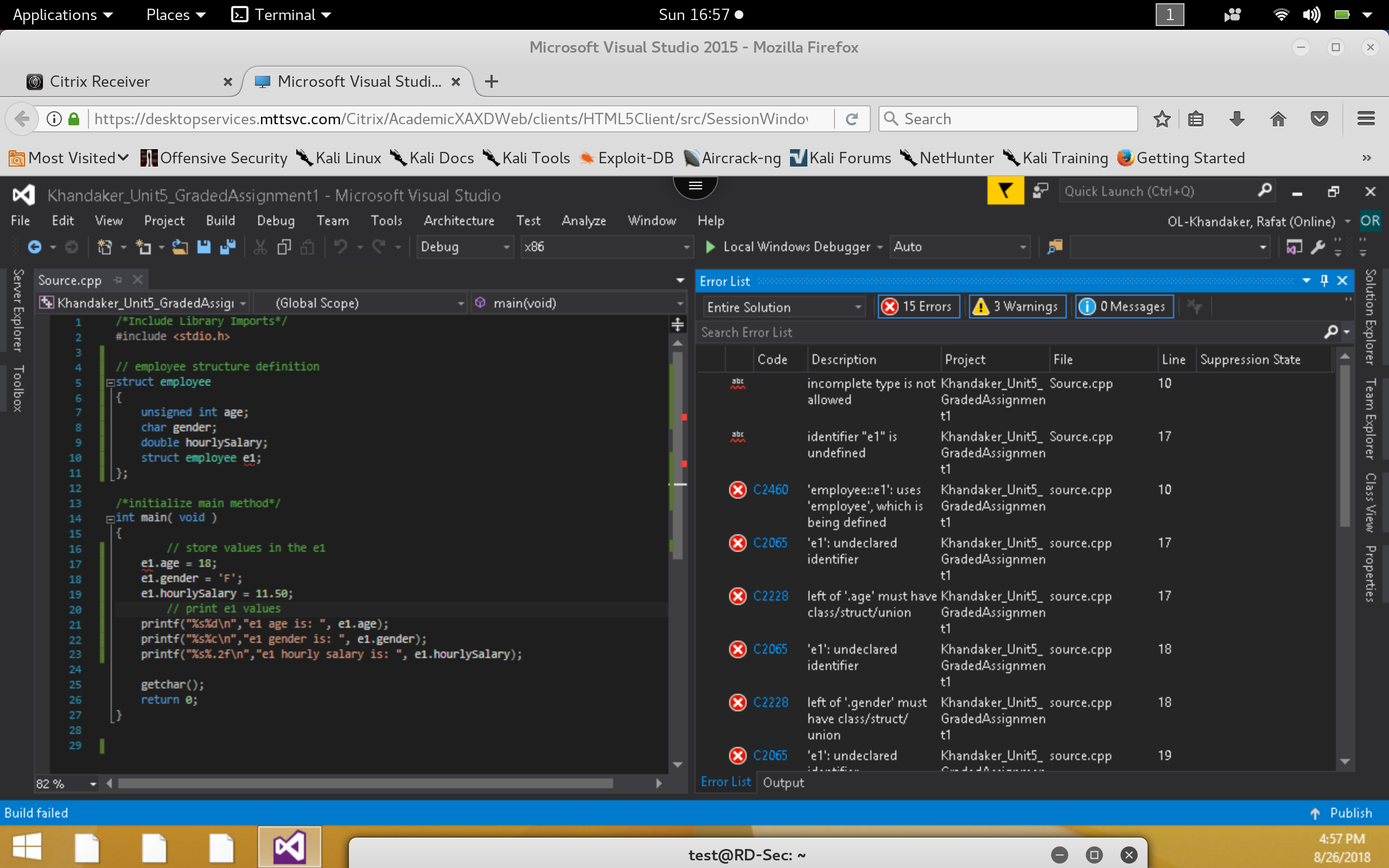


**Step 2**

1. **Execute the following code and identify the errors in the program. Debug the program and provide the correct version of the code.**

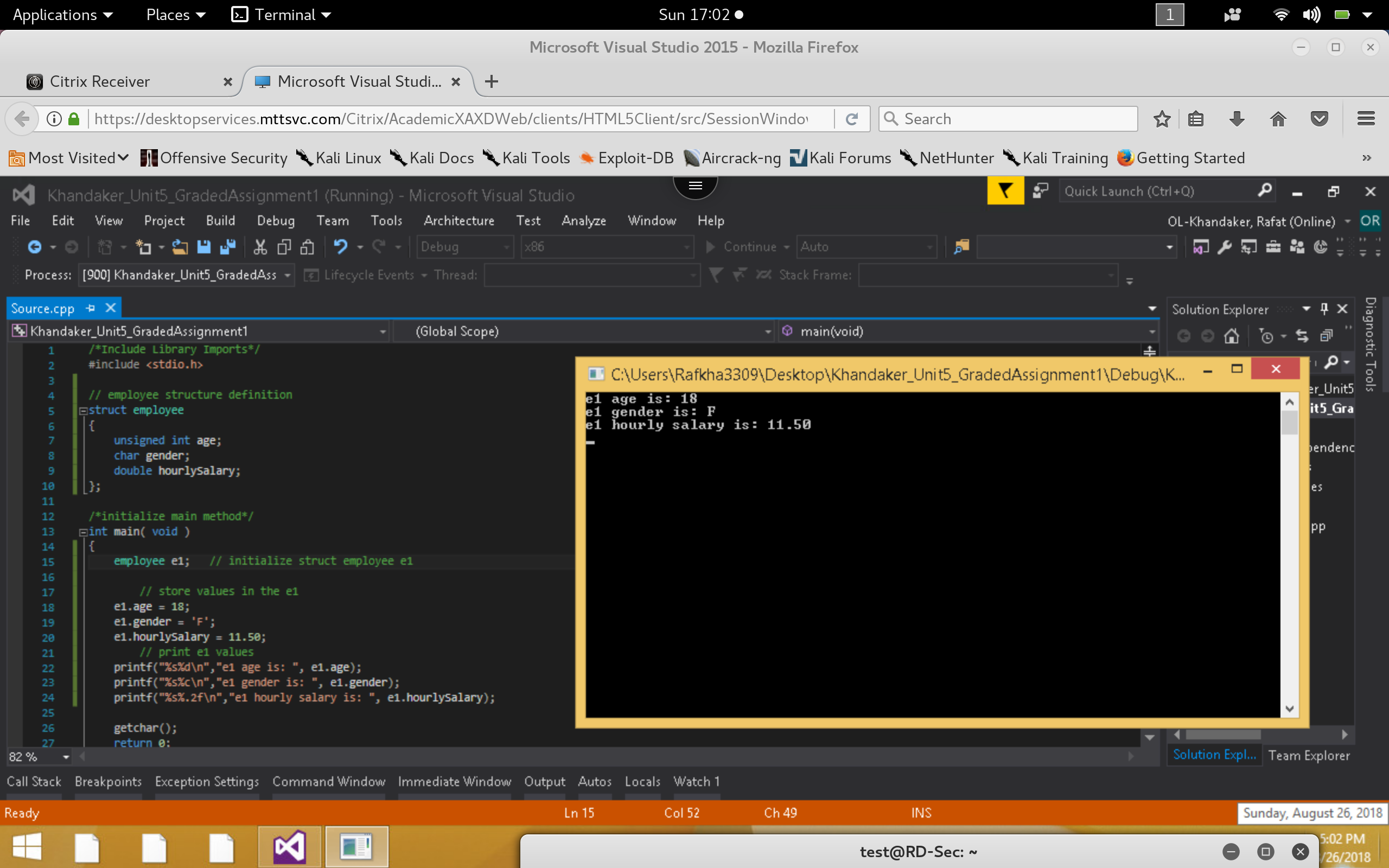
***Incorrect*:**

*In here, we are capturing 15 errors on runtime due to the initialization of the struct variable. We are not allowed to initialize a struct inside the definition of a struct.*



***Correct:***

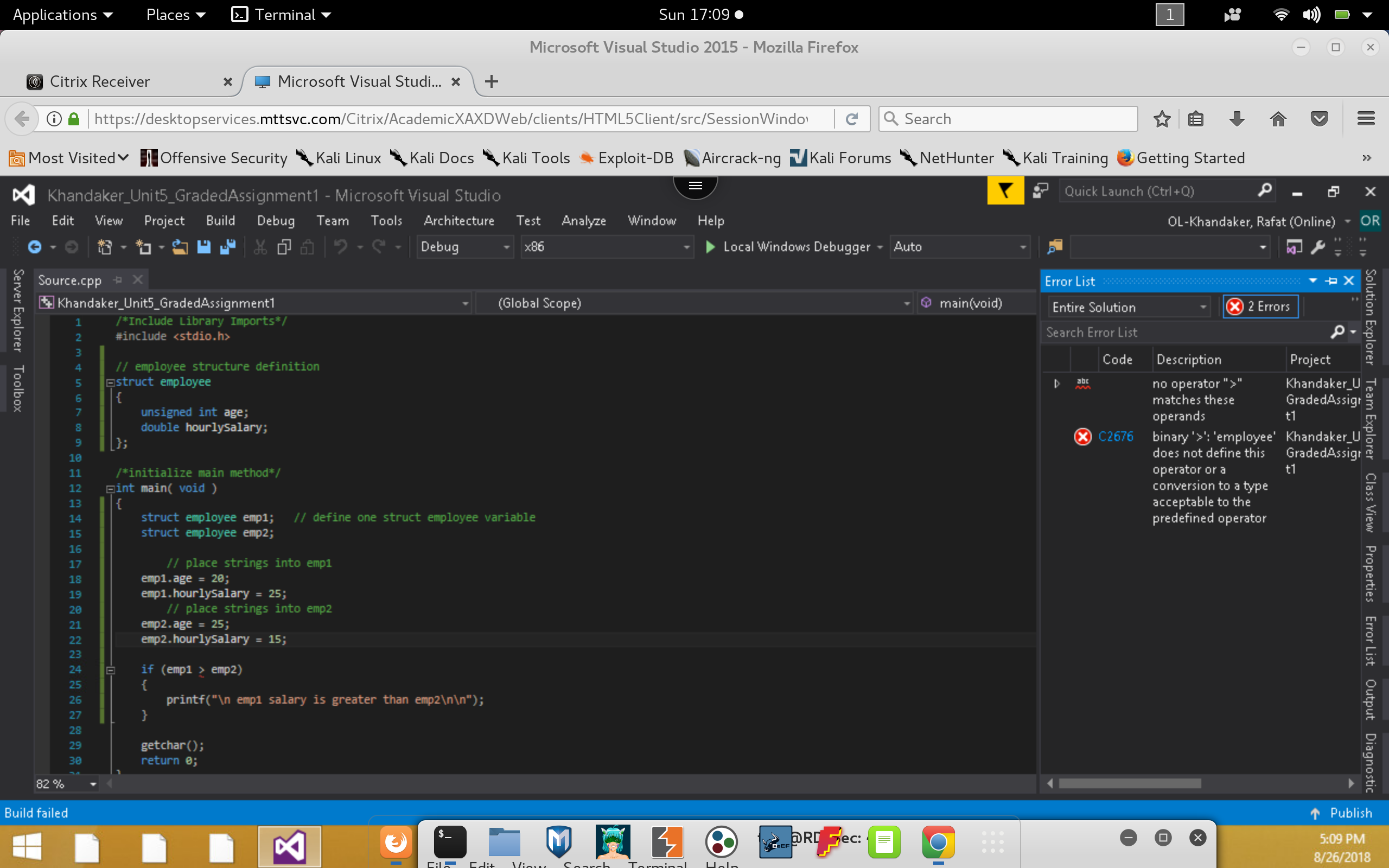
*To fix this error, we will simply initialize the e1 struct, inside the main method, which should remove the errors.*



1. **Execute the following code and identify the errors in the program. Debug the program and provide the correct version of the code.**

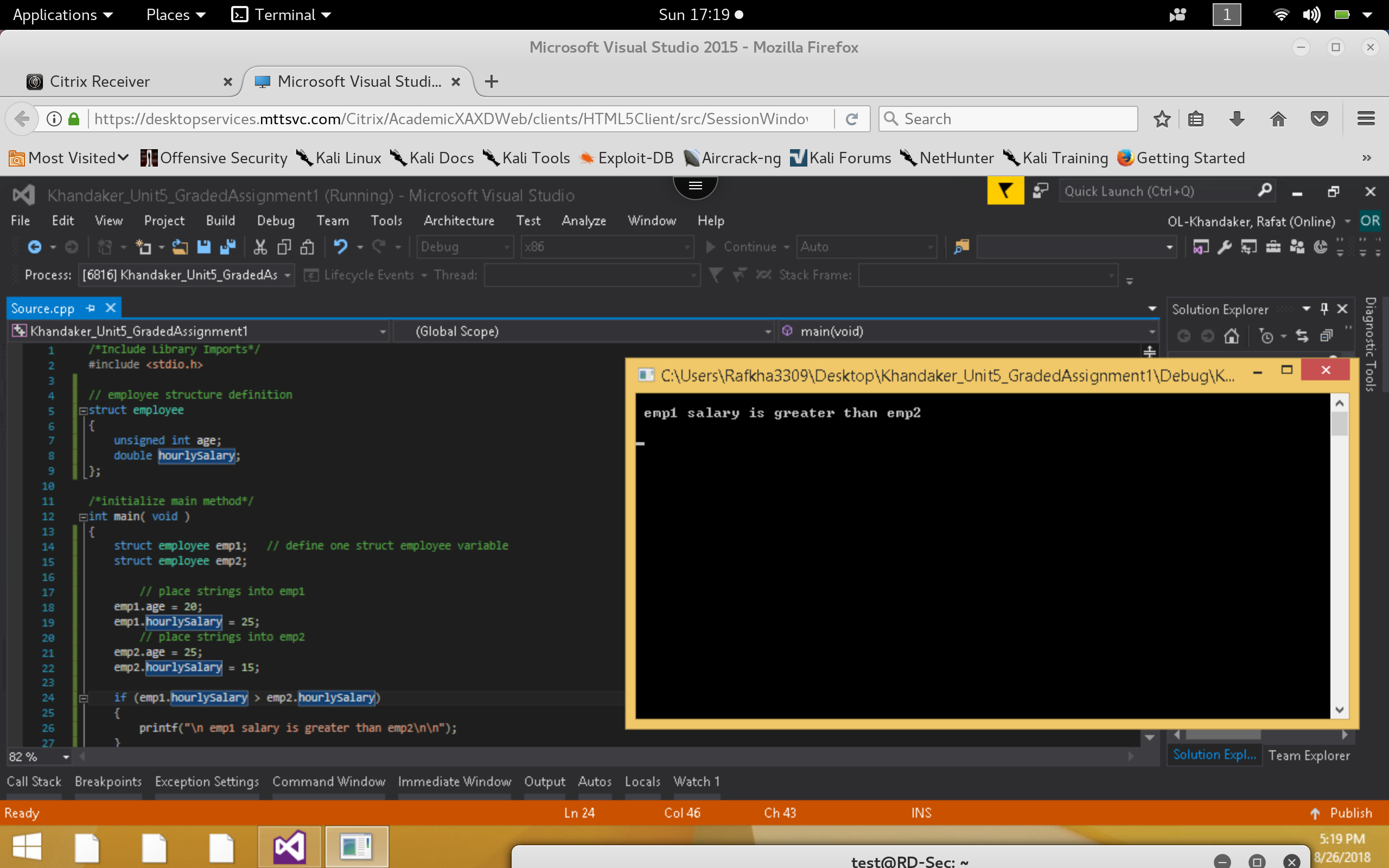
***Incorrect:***

*In this example, We capture an error at compile time near the greater than operator ‘>’This is because we cannot compare structs, directly. In order to do comparisons with operators, we must compare the values inside the struct individually.*



***Correct:***

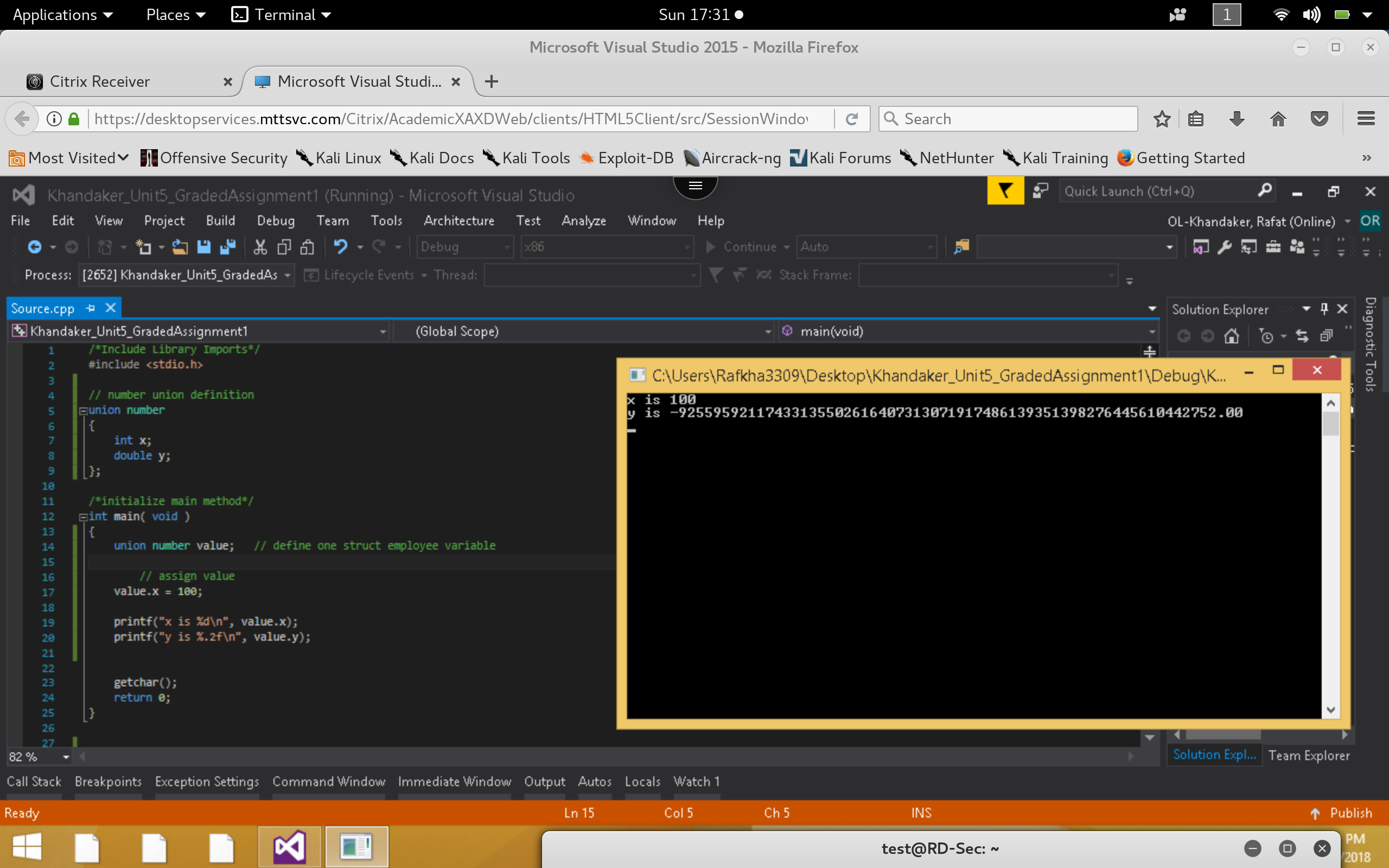
*To resolve this error, we simply compare the employees by hourly salary & then we get the proper output.*



1. **Execute the following code and identify the errors in the program. Debug the program and provide the correct version of the code. Note: Be sure to check the expected output to see if the correct values are being displayed. Hint: consider the purpose of a union.**

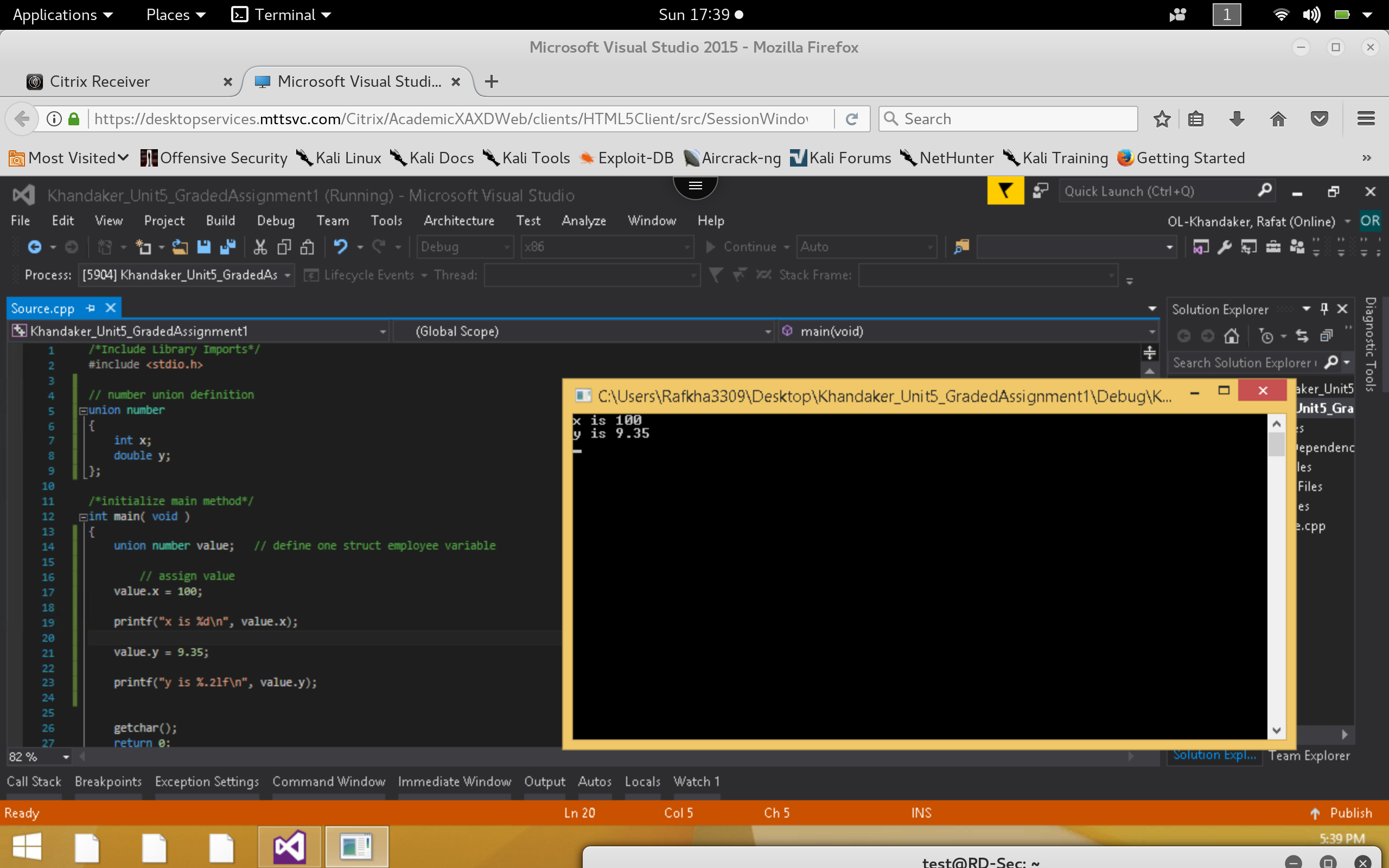
***Incorrect:***

*In this example, we get the incorrect output of value.y . In a union, we cannot assign multiple value to it’s memory address & expect the same result. This is because a union will try to re-write the memory with the last variable format, this will alter the value in the previous variable storage.*



***Correct:***

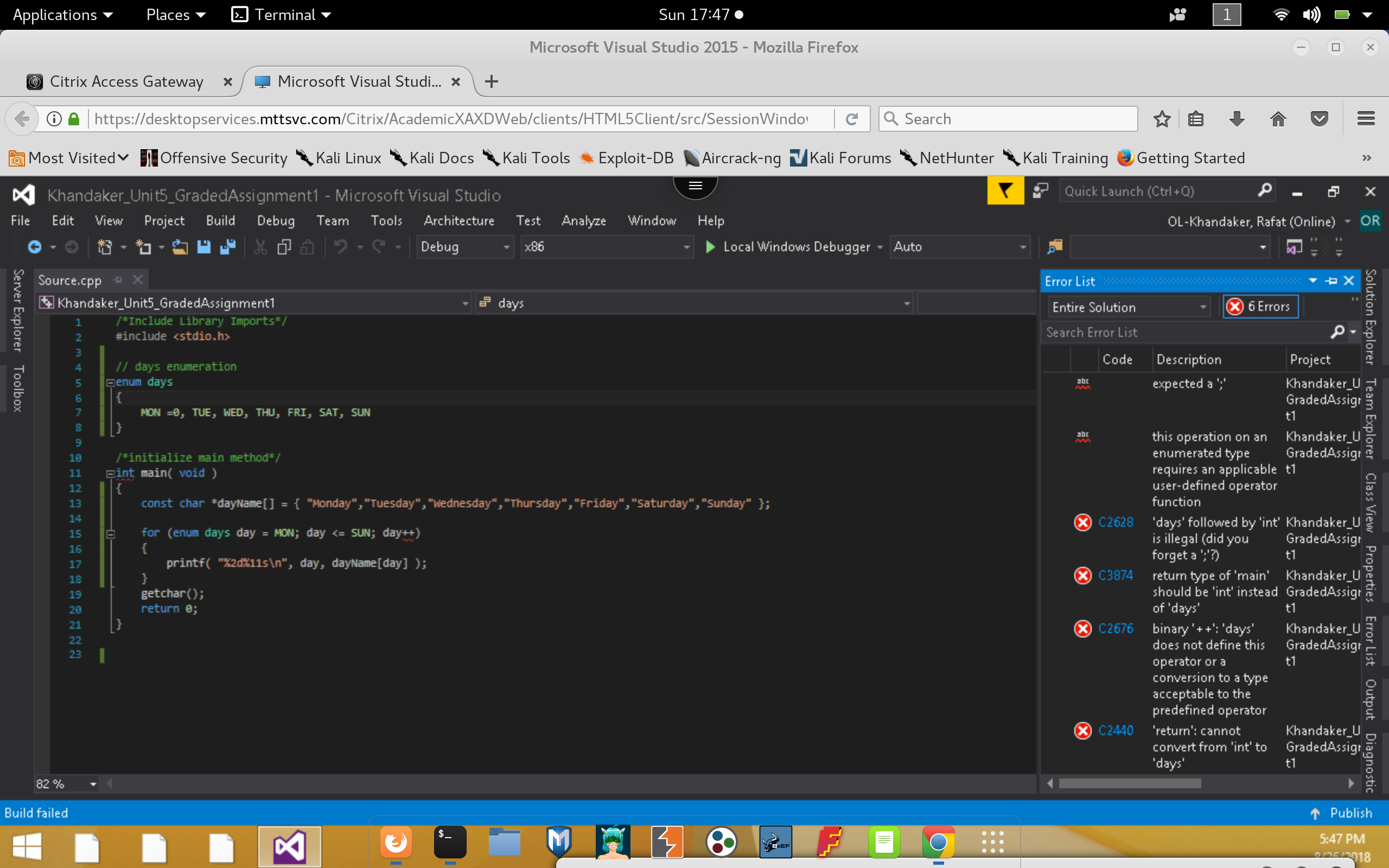
*To fix this problem, we must assign a value to y between the outputs, so the value is not over-written by the previous.*



1. **Execute the following code and identify the errors in the program. Debug the program and provide the correct version of the code.**

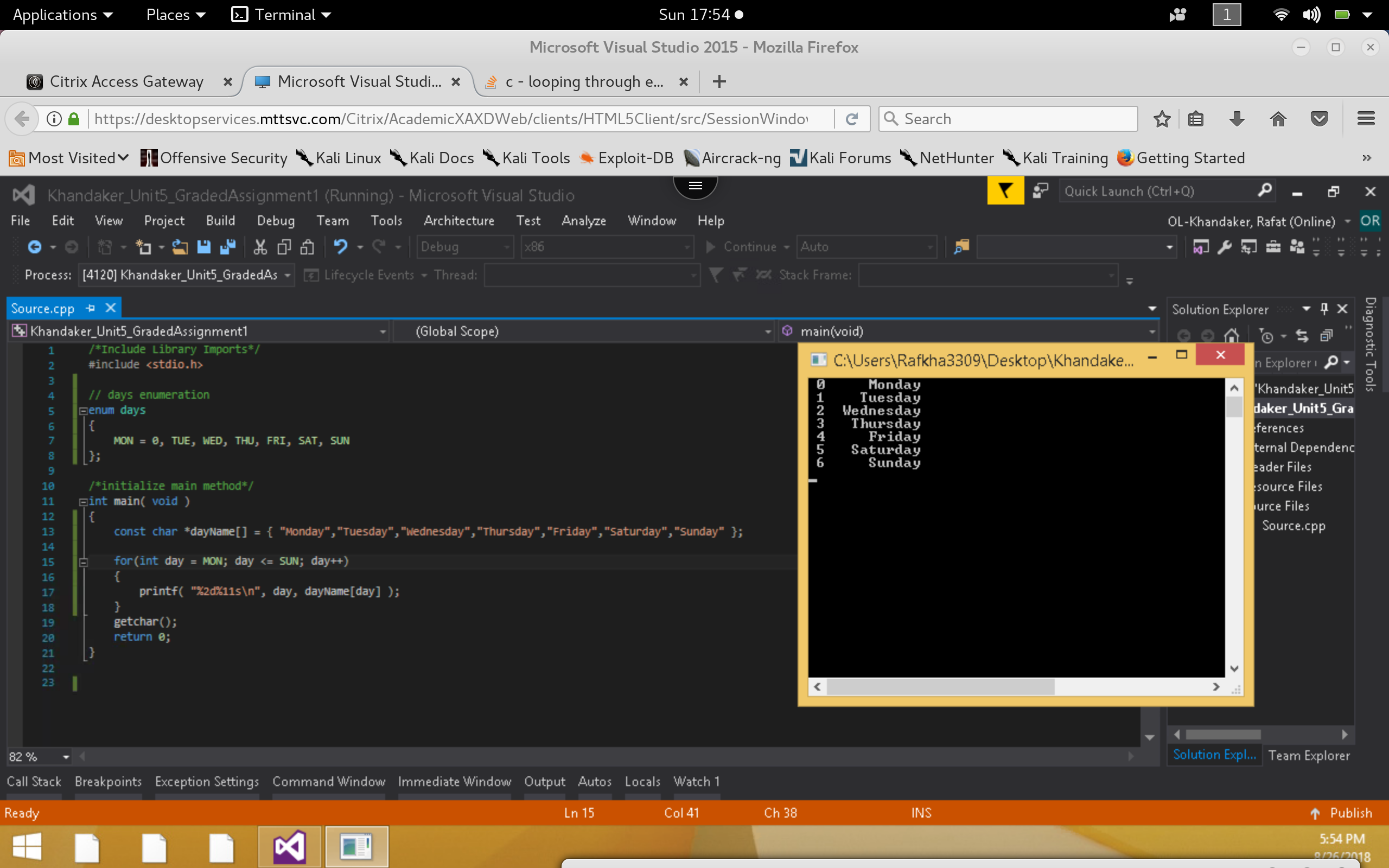
***Incorrect:***

*In this example, we retrieve the compile time errors during build process because we did not break the enumeration days with a semi-colon ‘;’ . We also need to initialize day as an int to be able to increment & iterate through the loop.*



***Correct:***

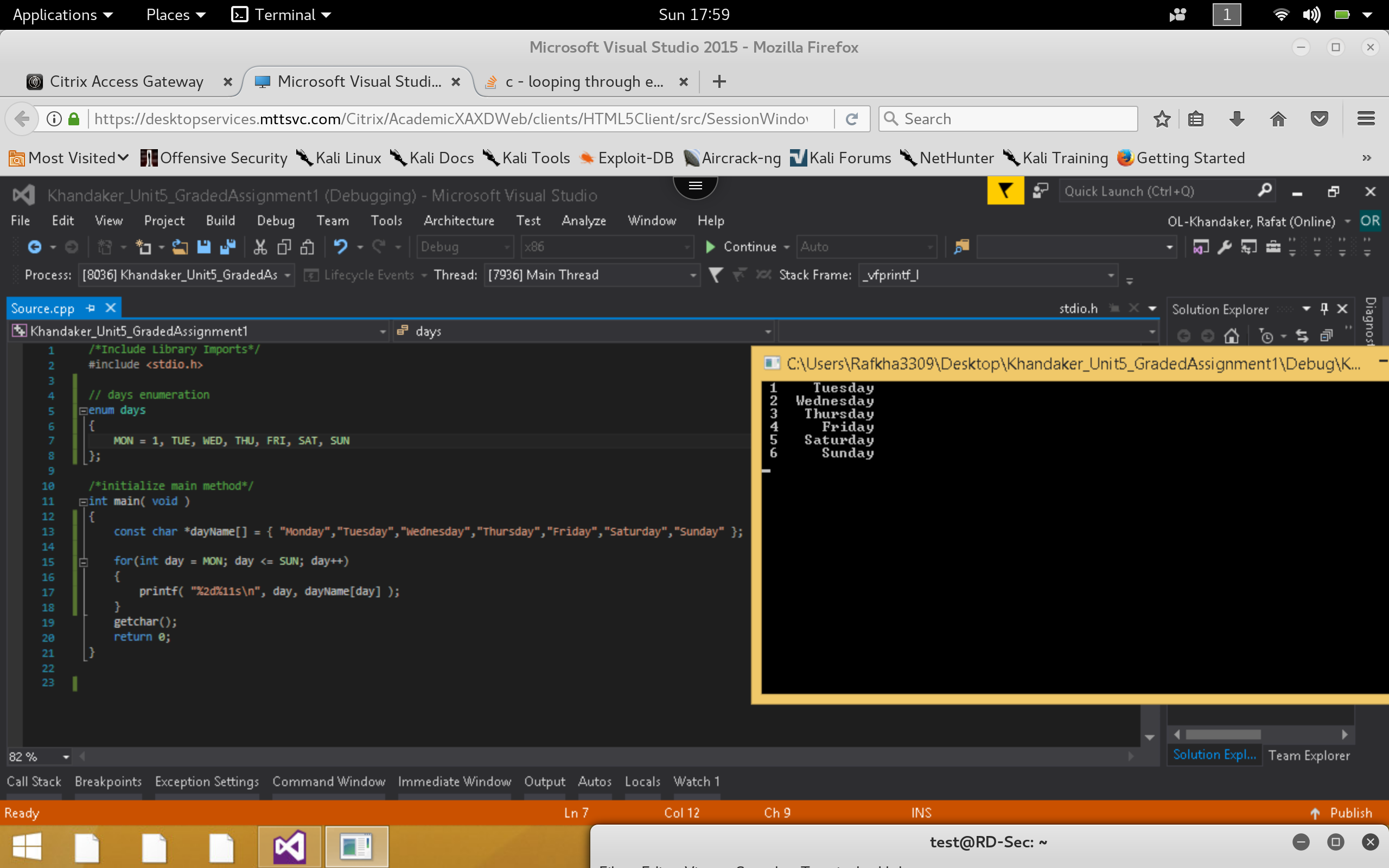
*To fix these issues, we will simply add a semi-colon to terminate the enum definition & initialize MON as an int value, to allow loop to iterate.*



1. **Execute the following code and identify the errors in the program. Debug the program and provide the correct version of the code.**

***Incorrect*:**

*In this example, We will iterate through an index out of bound exception because we initialized the value of MON to be 1. Our array value dayName[] will start at 0. So currently, our compiler prints the value from tuesday to sunday before break on exception is caught*



***Correct:***

*To fix this example, I will simply subtract 1 from the Day Name array index to retrieve the correct output.*

