CEng2xx Laboratory System User Manual

Introduction

2xx Laboratory system is an integrated environment for on-line quiz, program development and evaluation. During the laboratories you will have short exams measuring your knowledge and skills about programming. This system provides automatic execution of such examinations.

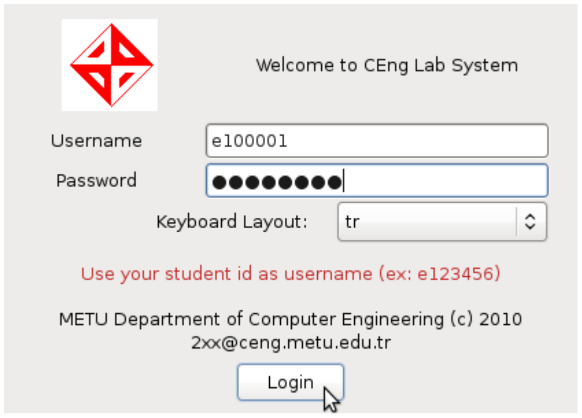
There are two basic parts of a laboratory session: (1) quiz, and (2) programming problem. In the quiz part you will take some multiple choice questions about the course. You will have a limited time and you should carefully answer the questions since wrong answers will cancel some portion of the correct answers as it is in most multiple choice tests.

After finishing the quiz part, you will be given a programming problem like “write a program to input ..., calculate and output ...”. In the remaining part of the laboratory session you will write a computer program described in the problem definition. A compiler environment is provided in the system. After you complete your program, you can test if your program is working correctly by choosing a menu option. After you are satisfied with the grade you are to take, you can finish the laboratory by finalizing your work.

This document describes the user interface and the steps that you should take during the laboratory.

Logging in

You are authenticated by a user name and password pair in the laboratories. Your grading and attendance are based on this authentication information. At the beginning of the laboratory, you will see the following window on your computers screen:


Welcome to CEng Lab System

Username: e100001

Password: ●●●●●●●●●

Keyboard Layout: tr

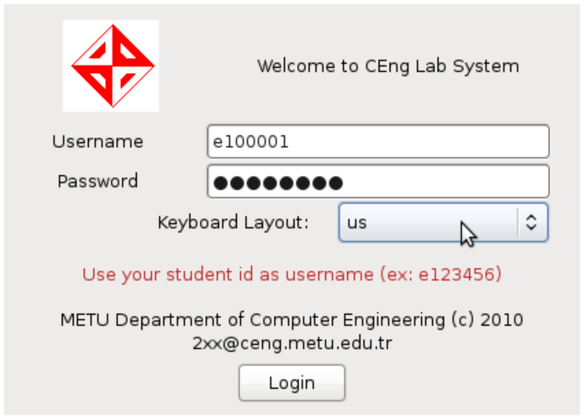
Use your student id as username (ex: e123456)

METU Department of Computer Engineering (c) 2010
2xx@ceng.metu.edu.tr

You should write your user identifier and password in the corresponding fields and press the `Login` button. Your login information will be sent to your METU mail accounts before your first laboratory session.

If you take a login failure message after you try and you are sure that you entered it correctly, ask help from your assistant.

You can change the keyboard layout to `tr` or `us`. When you change the keyboard layout on the Login Screen, it will be effective for the whole lab session.


Welcome to CEng Lab System

Username: e100001

Password: ●●●●●●●●

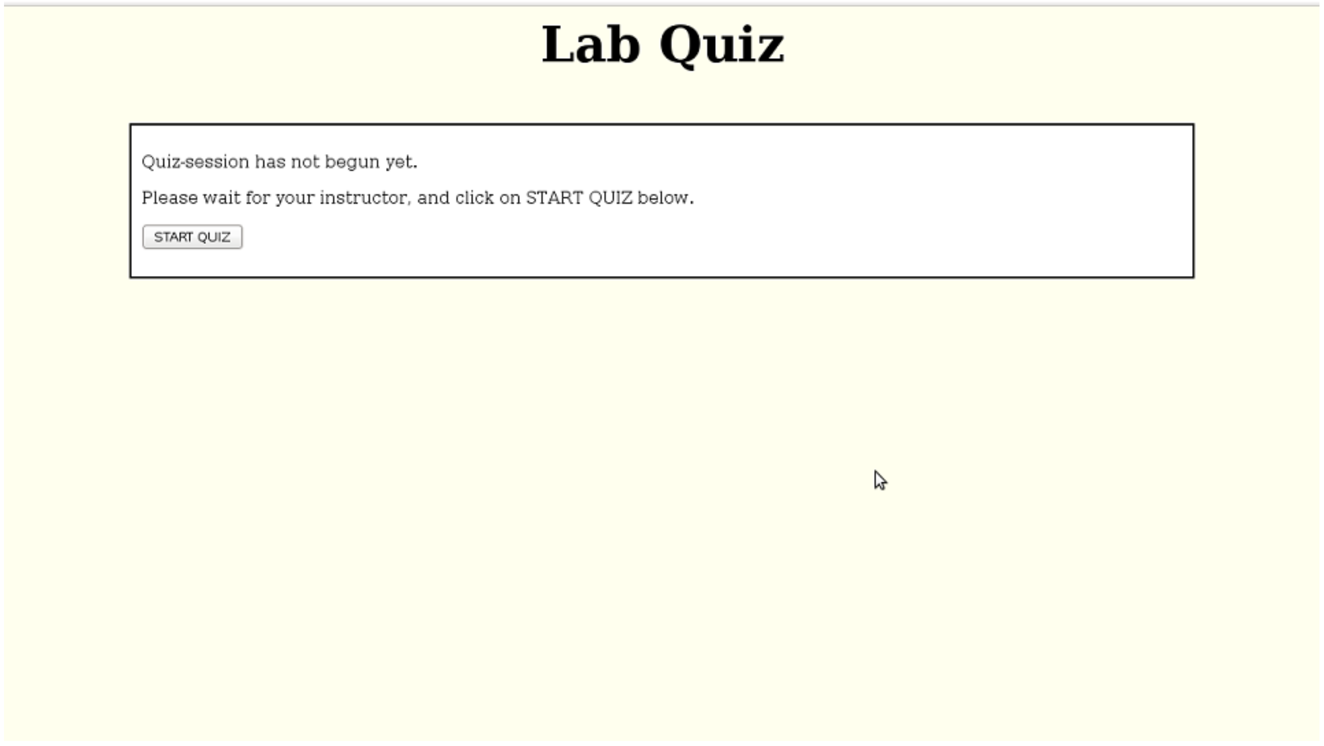
Keyboard Layout: us

Use your student id as username (ex: e123456)

METU Department of Computer Engineering (c) 2010
2xx@ceng.metu.edu.tr

Quiz

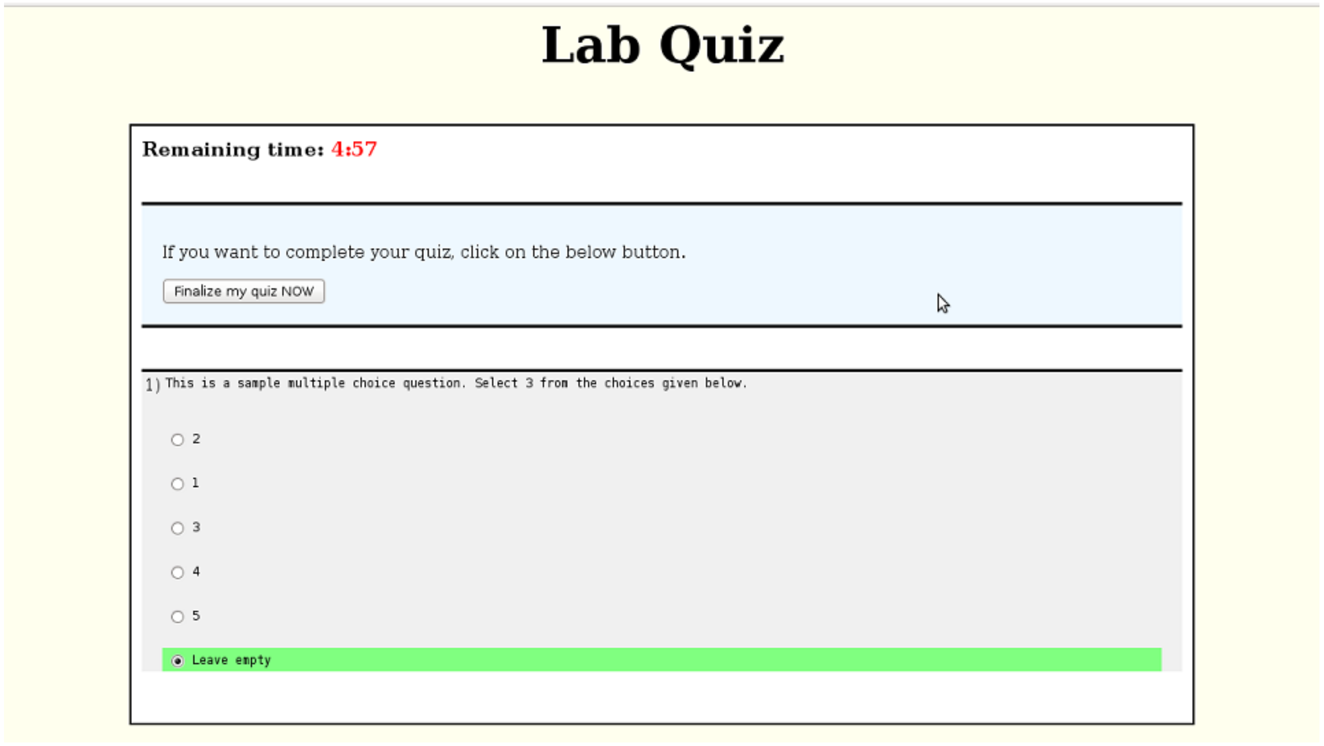
After you login, you will see the following screen


Lab Quiz

Quiz-session has not begun yet.

Please wait for your instructor, and click on START QUIZ below.

When your assistant starts the lab quiz, you can click on `Start Quiz` to start the quiz.



Lab Quiz

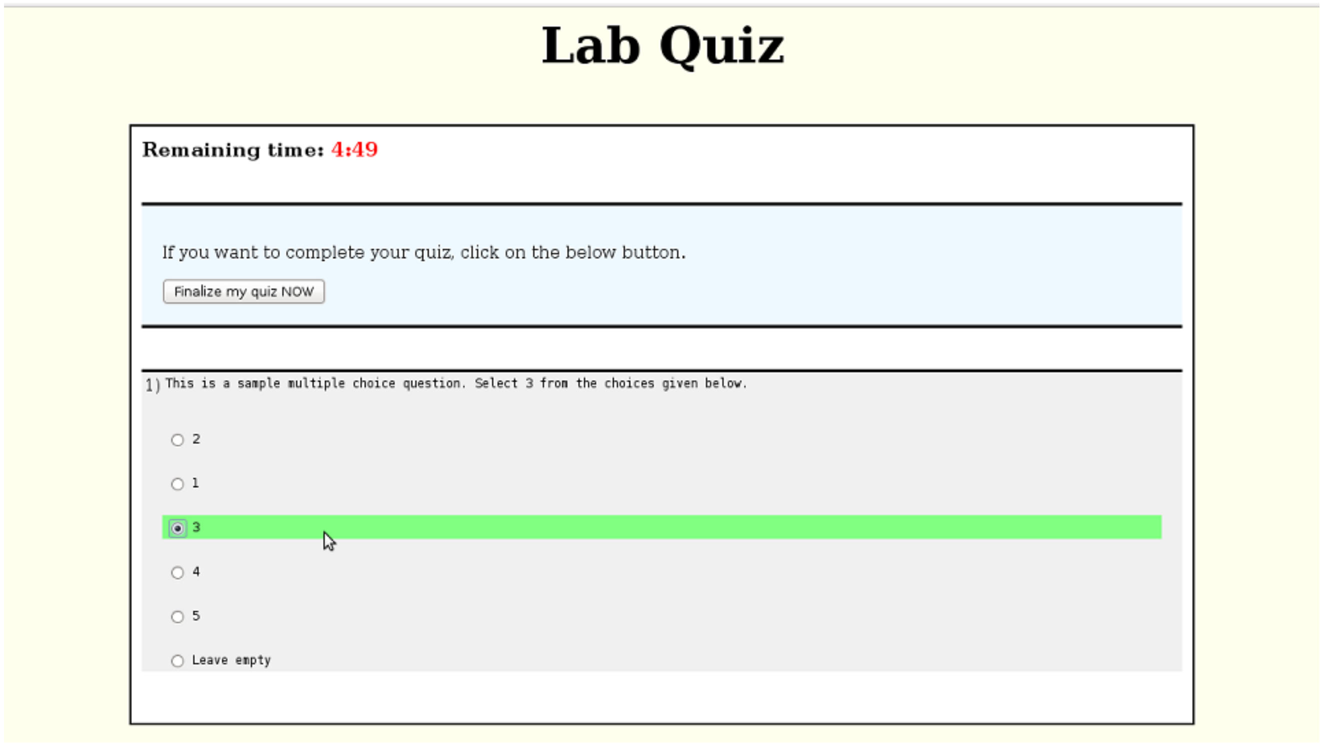
Remaining time: 4:57

If you want to complete your quiz, click on the below button.

1) This is a sample multiple choice question. Select 3 from the choices given below.

- ○ 2
- ○ 1
- ○ 3
- ○ 4
- ○ 5
- ● Leave empty

Quiz questions will be about the topic of the current week of laboratories. You will choose among the options below the quiz question. The option you selected will be colored in green. If you do not know the answer of the question, you can choose `Leave Empty` in order to leave the question empty so that you will not loose any points from a wrong answer.



Lab Quiz

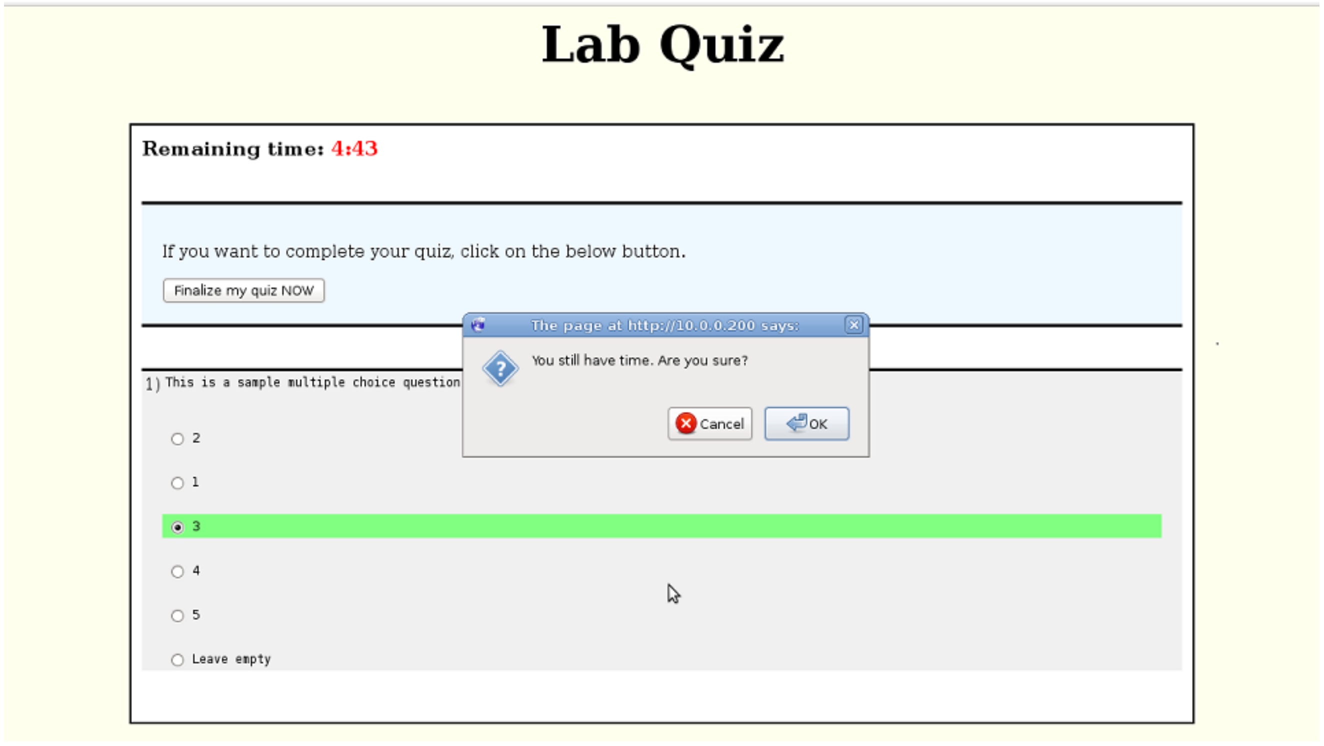
Remaining time: 4:49

If you want to complete your quiz, click on the below button.

1) This is a sample multiple choice question. Select 3 from the choices given below.

- ○ 2
- ○ 1
- ● 3
- ○ 4
- ○ 5
- ○ Leave empty

Time is limited in quiz part so watch the clock at the top of the screen `-Remaining time-` which will show how many seconds you have left. When your time is up all your answers will be with their current state. After you finished with the questions you can press the button `Finalize my quiz NOW` at the top of the window. When you click on this button, you will get the following message:



Lab Quiz

Remaining time: 4:43

If you want to complete your quiz, click on the below button.

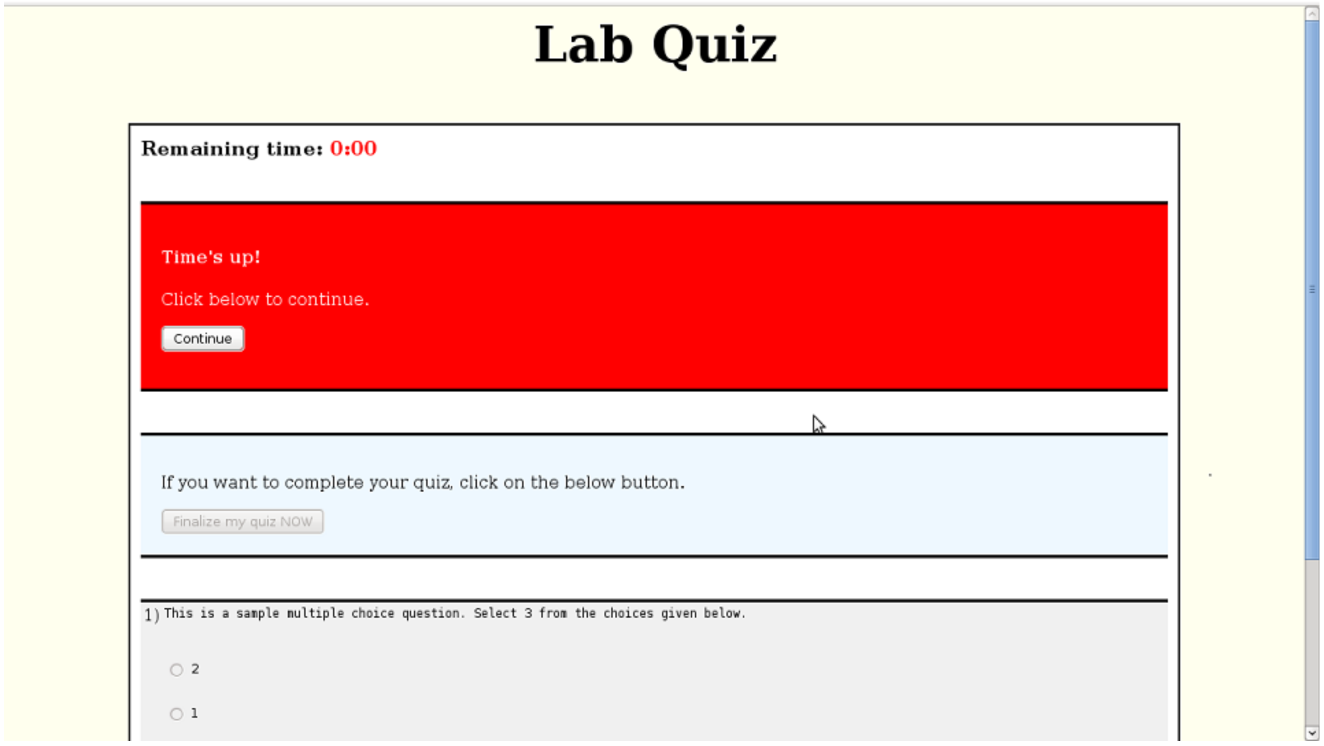
The page at http://10.0.0.200 says:

You still have time. Are you sure?

1) This is a sample multiple choice question

- ○ 2
- ○ 1
- ● 3
- ○ 4
- ○ 5
- ○ Leave empty

The following screen will appear, either if you finalized the quiz or if the quiz time has ended. Click `Continue` to continue with the programming part.



Lab Quiz

Remaining time: 0:00

Time's up!

Click below to continue.

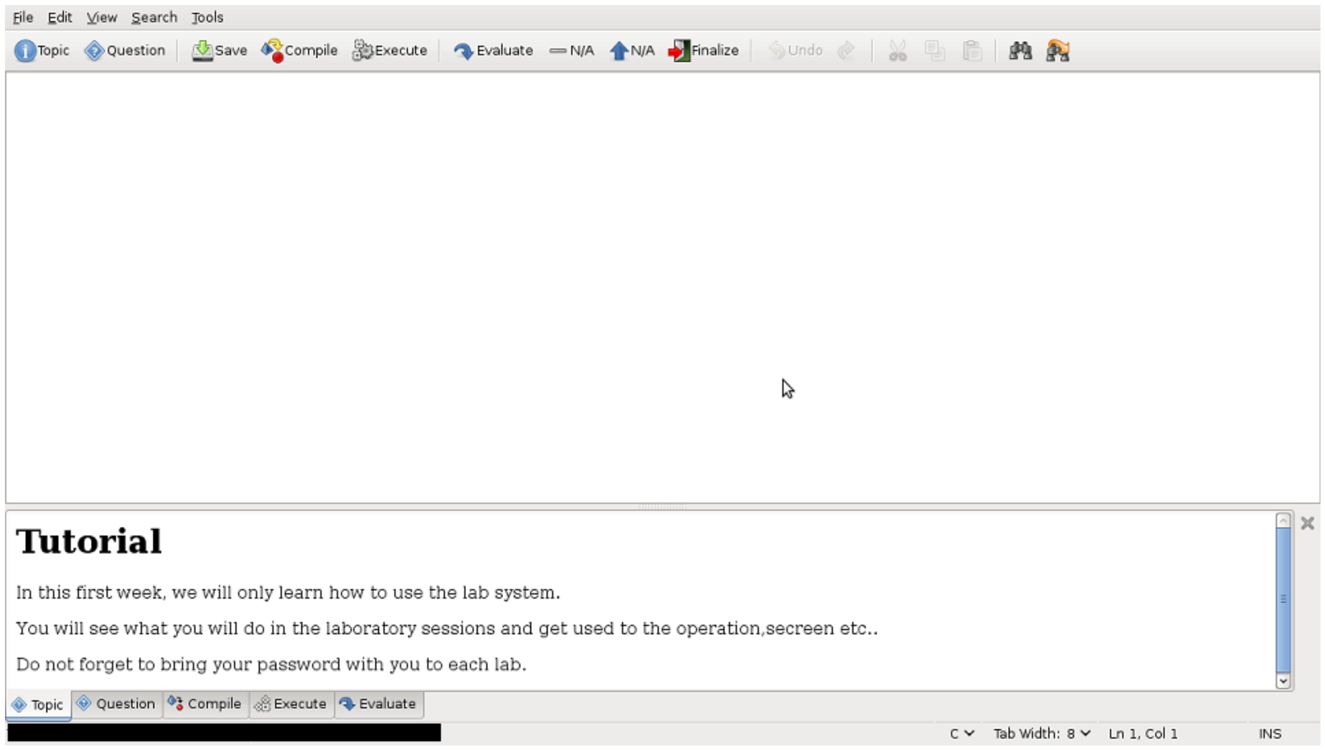
If you want to complete your quiz, click on the below button.

1) This is a sample multiple choice question. Select 3 from the choices given below.

- ○ 2
- ○ 1

Programming and Development Environment

Program development environment screen will look like:


File Edit View Search Tools
Topic Question Save Compile Execute Evaluate N/A N/A Finalize Undo
Topic Question Compile Execute Evaluate
C Tab Width: 8 Ln 1, Col 1 INS

Tutorial

In this first week, we will only learn how to use the lab system.

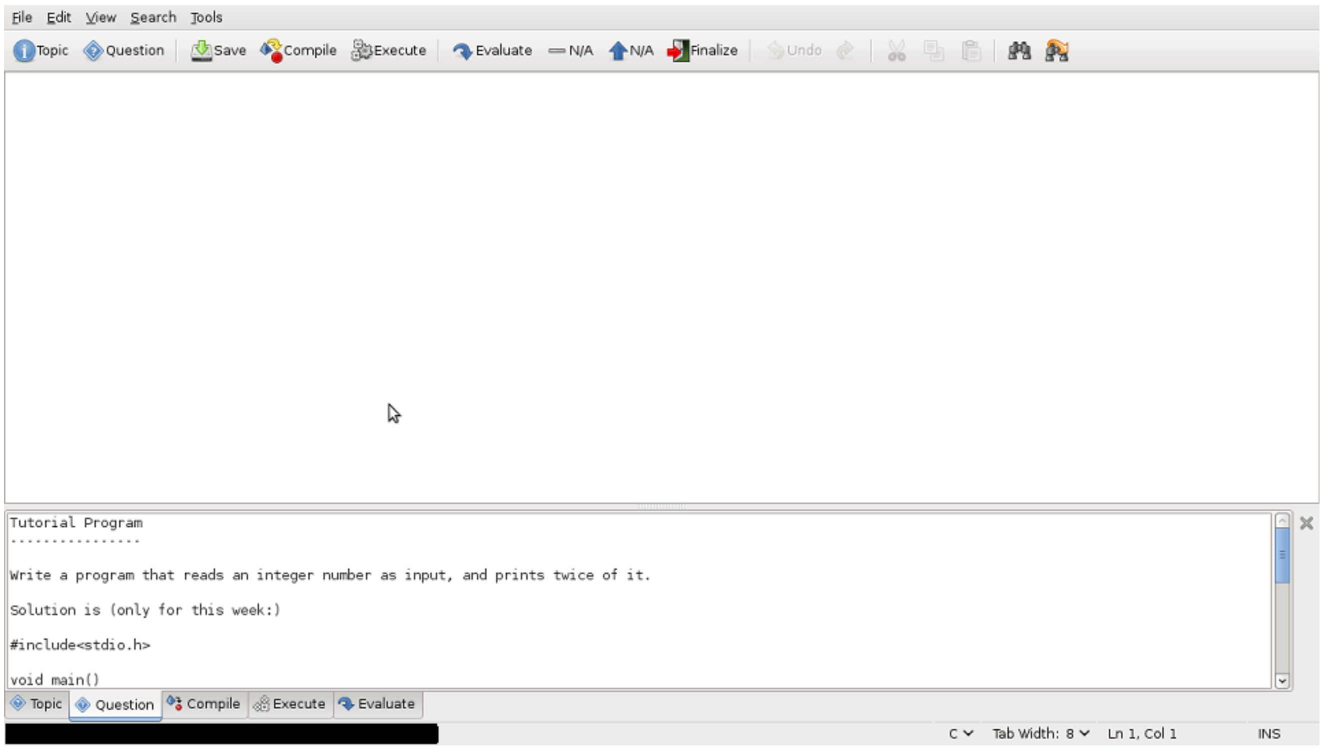
You will see what you will do in the laboratory sessions and get used to the operation,secreen etc..

Do not forget to bring your password with you to each lab.

At the top, there exists a *tool bar* with buttons "`Topic`", "`Question`", "`Save`", "`Compile`", "`Execute`", "`Evaluate`" and "`Finalize`". When one of these buttons are clicked, their output will be displayed on the bottom pane. You will write your programs on the middle text area. At the very bottom, there is a *message line* containing useful information such as the line and colon number of the cursor.

Programming Question

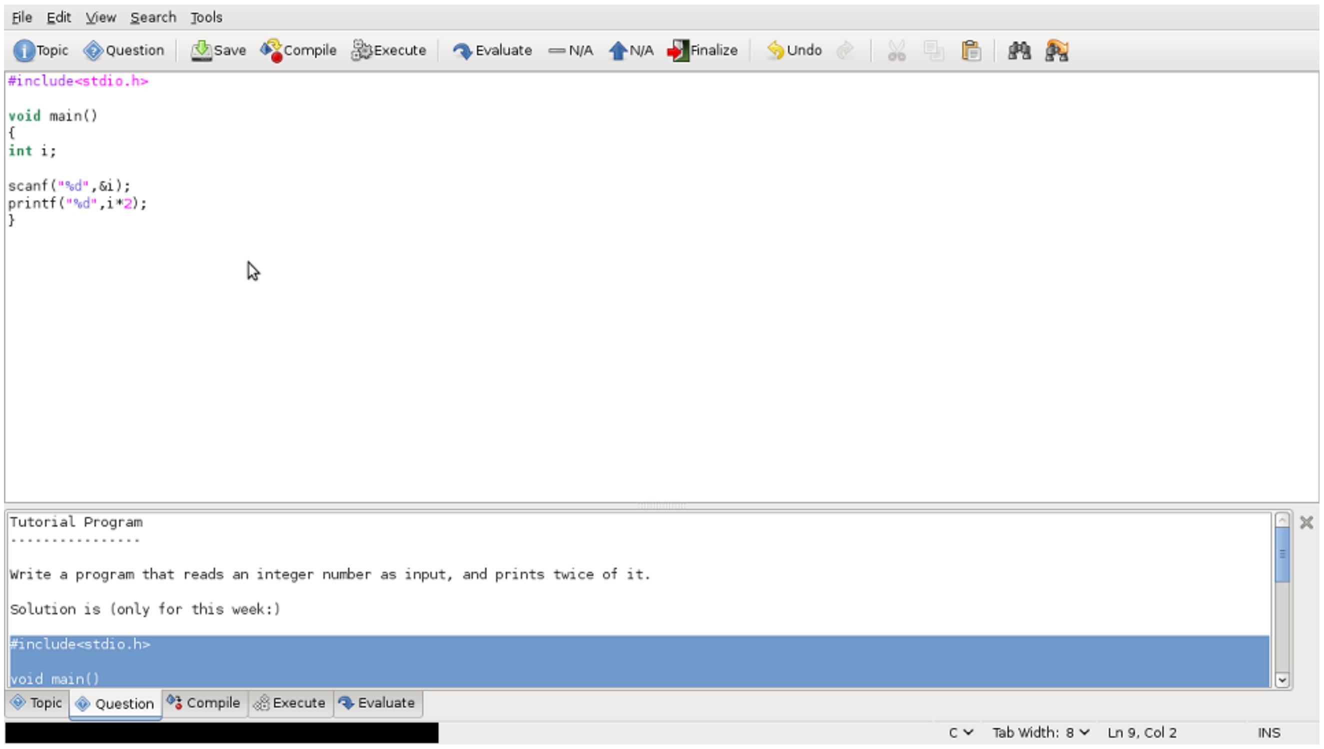
Click on `Question` button to see the programming question. The programming problem, its input and output requirements will be described in the message area. Your task is to write a complete program in the editing window to fulfill this description.


File Edit View Search Tools
Topic Question Save Compile Execute Evaluate N/A N/A Finalize Undo
Topic Question Compile Execute Evaluate
C Tab Width: 8 Ln 1, Col 1 INS

```
Tutorial Program
----------------

Write a program that reads an integer number as input, and prints twice of it.

Solution is (only for this week:)

#include<stdio.h>

void main()
```

[redacted]

Write your code in the text area provided in the middle of the screen.


File Edit View Search Tools
Topic Question Save Compile Execute Evaluate N/A N/A Finalize Undo
Topic Question Compile Execute Evaluate
C Tab Width: 8 Ln 9, Col 2 INS

```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i);
printf("%d",i*2);
}
```

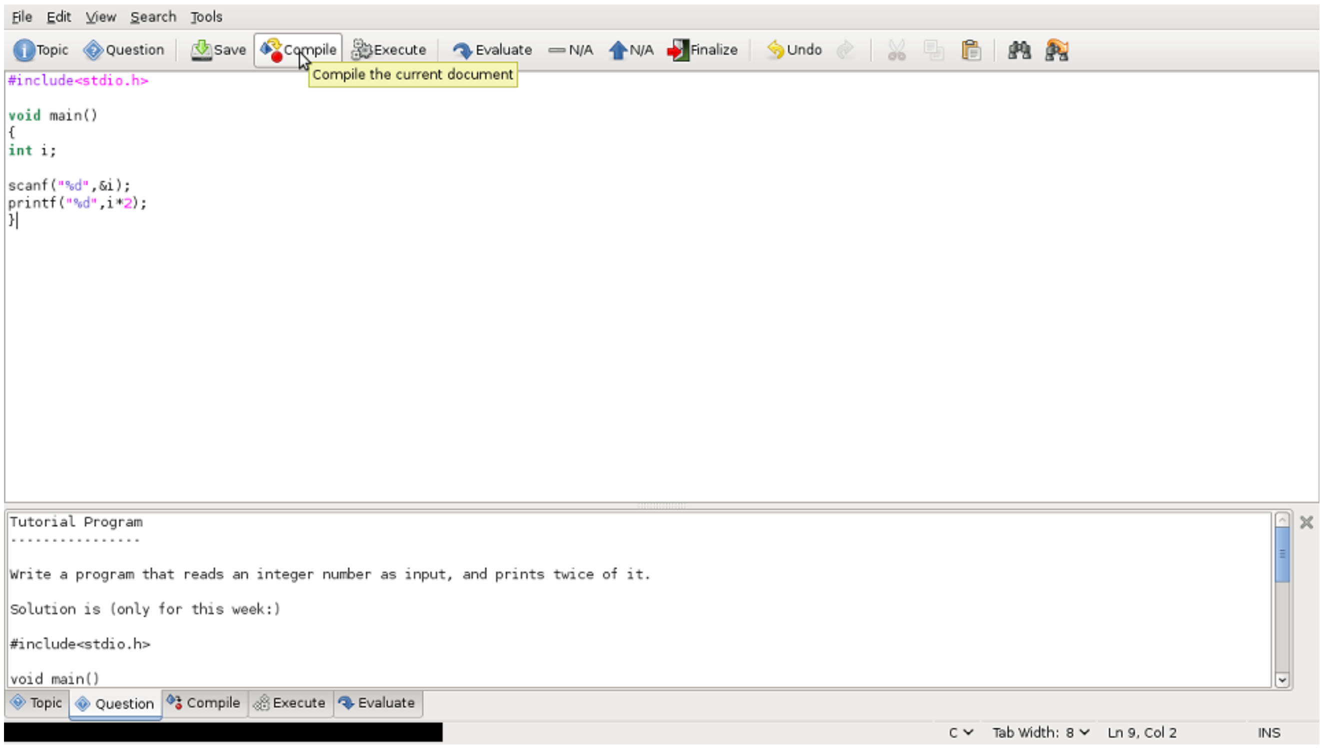
```
Tutorial Program
----------------

Write a program that reads an integer number as input, and prints twice of it.

Solution is (only for this week:)

#include<stdio.h>

void main()
```

[redacted]

Compiling Your program

After reading laboratory topic and question text carefully you can start writing your program. Make sure that you are writing a full and syntactically correct program. For example you should include standard IO header file and put a correct `main()` function body in a C program.

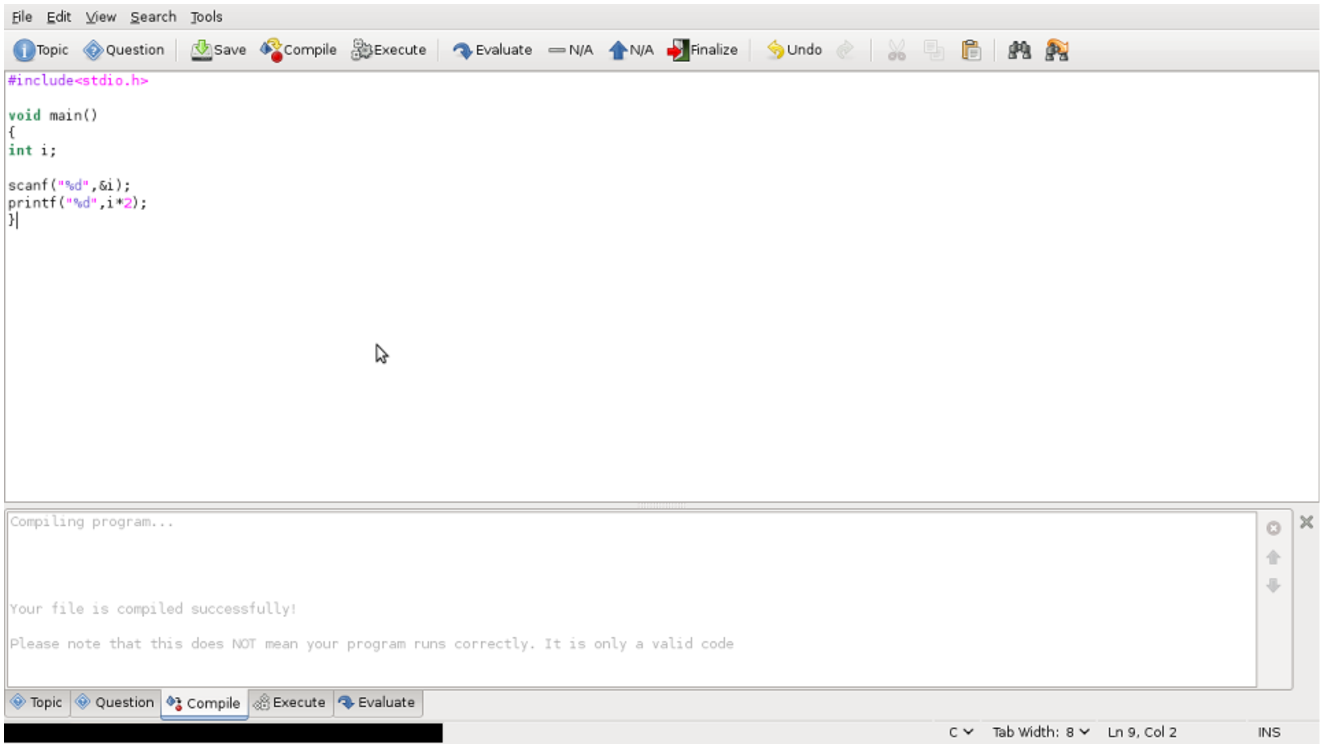
When your program is ready to compile, press the `Compile` button to compile it. Your computer will compile the code residing in the editing area and report you the compilation result in the message area. If it is successful you can go on testing your program by running it. However in the first trial probably your program will fail to compile because of compilation errors. Do not worry about it. Compiler provide the line of the error and other useful information in the message area. Go there, find and correct your errors. A detailed description of the compilation errors and possible correction strategies are given in the appendix.


File Edit View Search Tools
Topic Question Save Compile Execute Evaluate N/A N/A Finalize Undo
Compile the current document
Topic Question Compile Execute Evaluate
C Tab Width: 8 Ln 9, Col 2 INS

```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i);
printf("%d",i*2);
}
```

```
Tutorial Program
----------------

Write a program that reads an integer number as input, and prints twice of it.

Solution is (only for this week:)

#include<stdio.h>

void main()
```

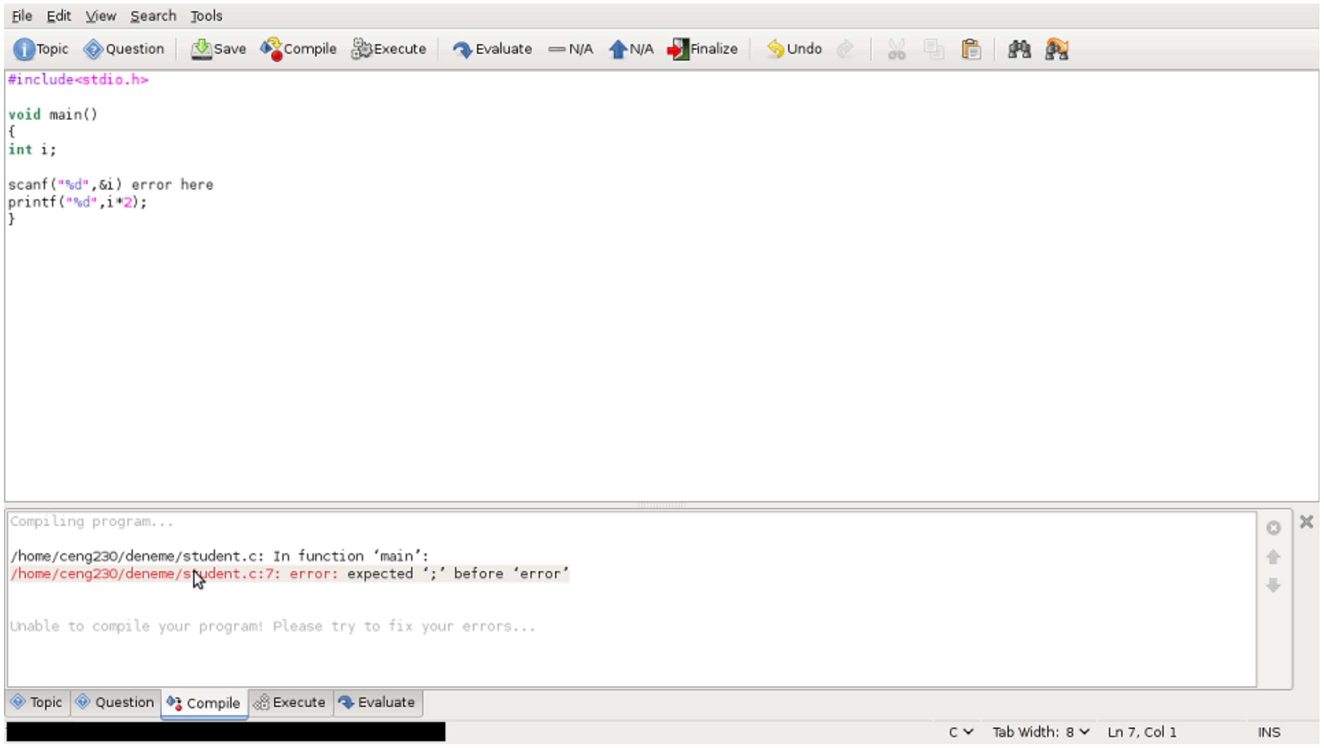
The compile output will be displayed in the bottom pane.


File Edit View Search Tools
Topic Question Save Compile Execute Evaluate N/A N/A Finalize Undo
Topic Question Compile Execute Evaluate
C Tab Width: 8 Ln 9, Col 2 INS

```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i);
printf("%d",i*2);
}
```

```
Compiling program...


Your file is compiled successfully!

Please note that this does NOT mean your program runs correctly. It is only a valid code
```

In case of compile errors, the output window will show the errors in color. You can click on the errors to go to the line on the text area that is causing the problem.

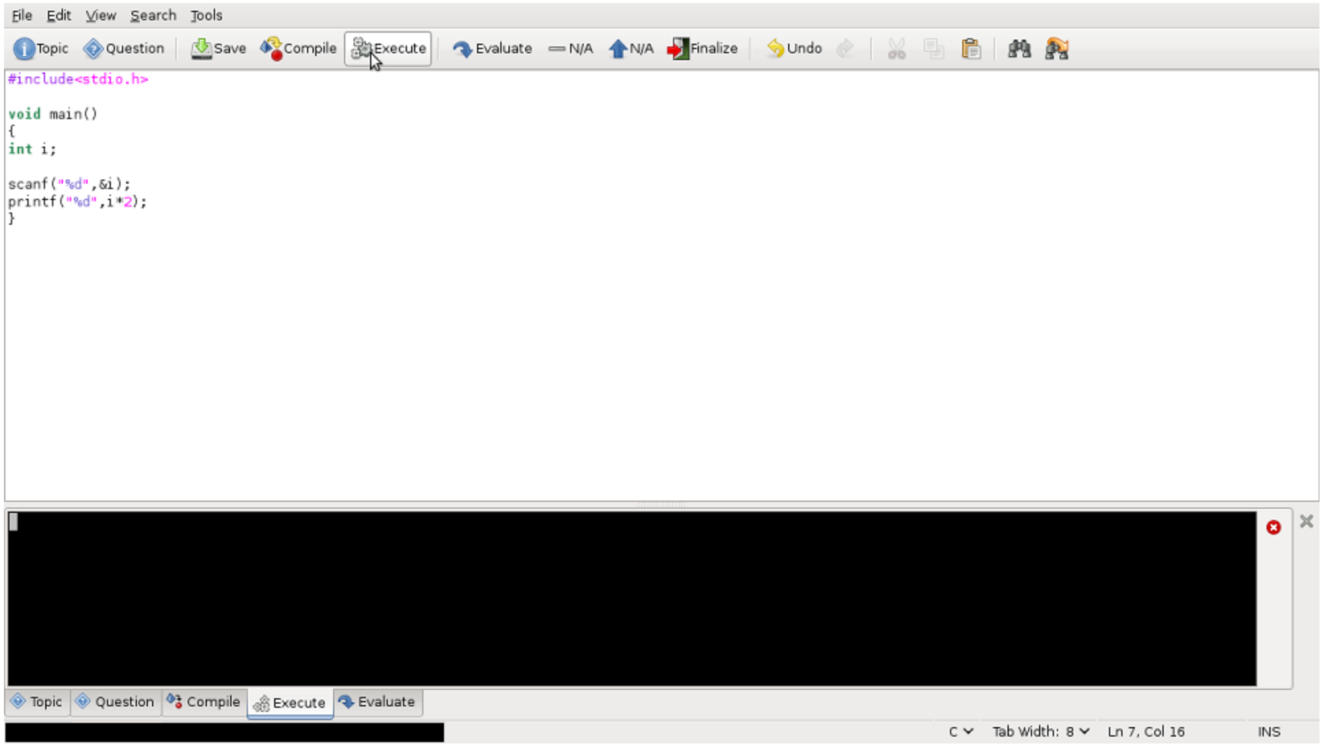

File Edit View Search Tools
Topic Question Save Compile Execute Evaluate N/A N/A Finalize Undo
Topic Question Compile Execute Evaluate
C Tab Width: 8 Ln 7, Col 1 INS

```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i) error here
printf("%d",i*2);
}
```

```
Compiling program...

/home/ceng230/deneme/student.c: In function ‘main’:
/home/ceng230/deneme/student.c:7: error: expected ‘;’ before ‘error’


Unable to compile your program! Please try to fix your errors...
```

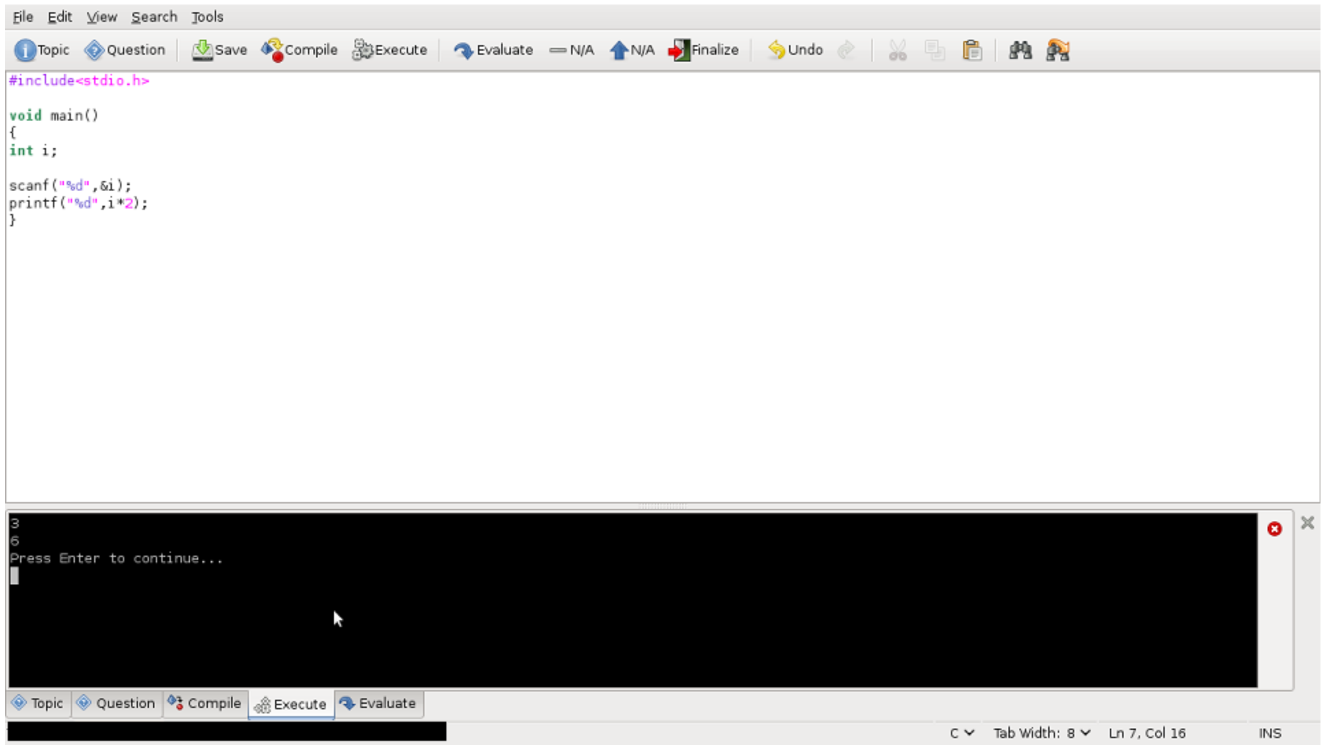
Running Your program

When your program compiles successfully, it is ready to be tested. You can run it by clicking on the `Execute`button in the menu bar.


File Edit View Search Tools
Topic Question Save Compile Execute Evaluate N/A N/A Finalize Undo
Topic Question Compile Execute Evaluate
C Tab Width: 8 Ln 7, Col 16 INS

```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i);
printf("%d",i*2);
}
```

You can give input to and read the output of the program from the bottom pane to test your code. During this test, try possible cases of the problem and observe the reaction of your program. This information is very important while you are tracking your algorithmic mistakes. When you think your program is ready, you can evaluate it.


File Edit View Search Tools
Topic Question Save Compile Execute Evaluate N/A N/A Finalize Undo
Topic Question Compile Execute Evaluate
C Tab Width: 8 Ln 7, Col 16 INS

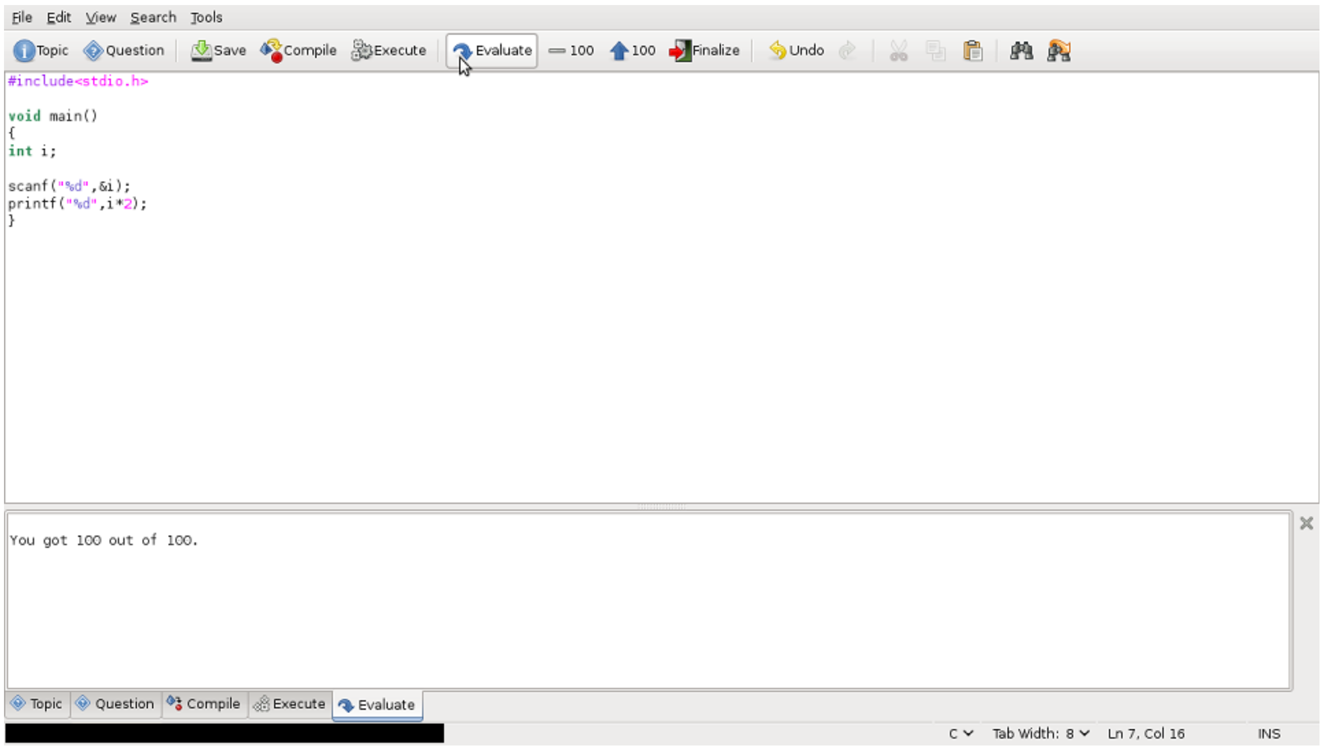
```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i);
printf("%d",i*2);
}
```

```
3
6
Press Enter to continue...
```

Evaluating Your Program

If you think your code is correct, you can click on the `Evaluate` button to evaluate your program.

When you press the `Evaluate` button, your program goes under an automatic evaluation process so that you will see the grade that you will take if you finalize it.


File Edit View Search Tools
Topic Question Save Compile Execute Evaluate 100 100 Finalize Undo
Topic Question Compile Execute Evaluate
[redacted]
C Tab Width: 8 Ln 7, Col 16 INS

```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i);
printf("%d",i*2);
}
```

```
You got 100 out of 100.
```

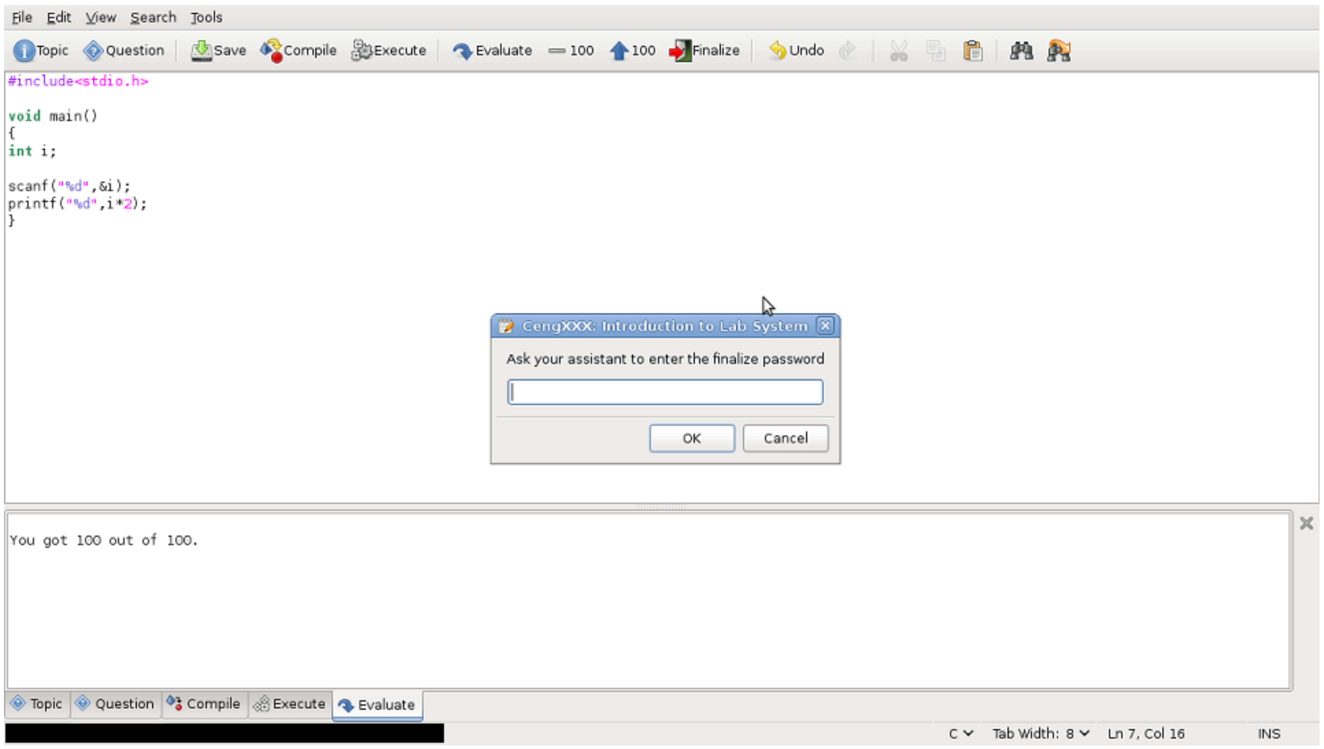
Message area on the bottom pane and the buttons on the toolbar display the grade that you would take if you finalize it. The button on the right of the `Evaluate` button shows your latest grade and the button next to it shows the highest grade you got from the prior evaluations. Your highest grade will be the one that counts when the programming part is finalized. If the grade is satisfactory you can go on finalizing. Otherwise turn back to your program, try to find your mistakes, correct, compile, execute and evaluate it again.

It is not a good practice to evaluate your program to often. You loose to much time and after some time you will be dealing with irrelevant small mistakes and fail to see your mistakes in the greater picture. So test your programs by running them and giving inputs of different cases. This is a better way of learning the programming language.

Finalizing Your Program

After you get a satisfactory point, click the `Finalize` button to finalize the programming part. This will end up your session and <u>you will not be able return back to your program</u>. So this is the very last thing you should do at the end of the laboratory. You will see the following message dialog.

You should call your laboratory assistant to approve your finish. If you do not follow the specifications (for example, do not write a function although it is required), or if s/he suspects that you cheated, s/he will reject to finalize your work.



```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i);
printf("%d",i*2);
}
```

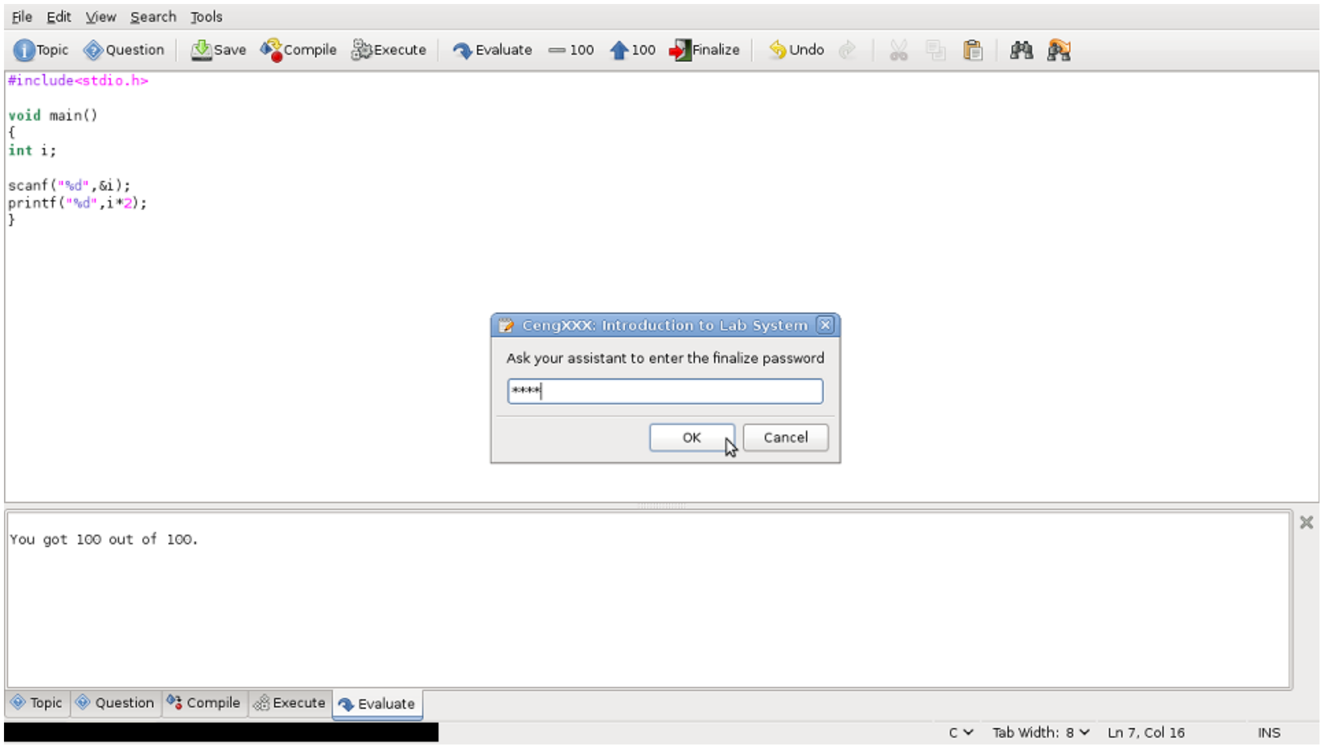
CengXXX: Introduction to Lab System

Ask your assistant to enter the finalize password

```
You got 100 out of 100.
```

C Tab Width: 8 Ln 7, Col 16 INS

After your laboratory assistant enters the confirmation password, click on the OK button.



```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i);
printf("%d",i*2);
}
```

CengXXX: Introduction to Lab System

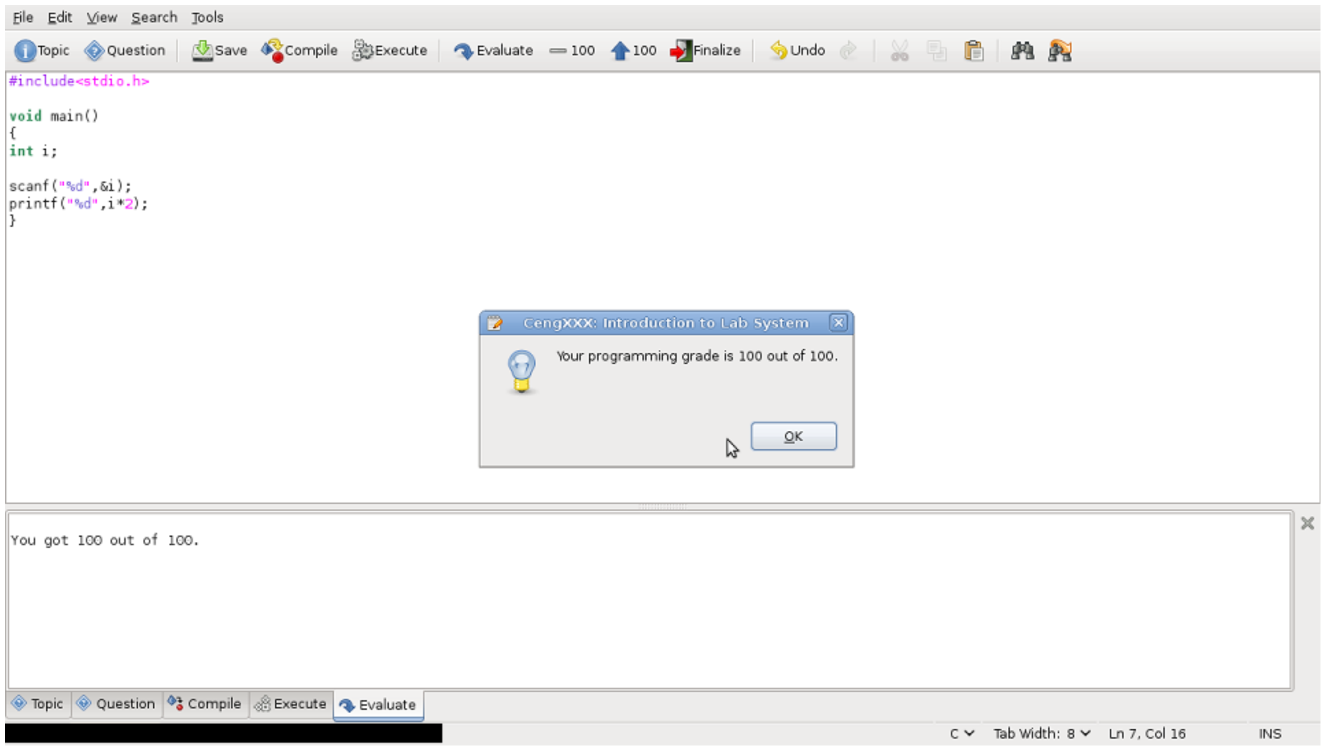
Ask your assistant to enter the finalize password

****

```
You got 100 out of 100.
```

C Tab Width: 8 Ln 7, Col 16 INS

Congratulations, the laboratory is over for you. You can leave the laboratory. Do not forget to click on the OK button to quit the program.


File Edit View Search Tools
Topic Question Save Compile Execute Evaluate 100 100 Finalize Undo
OK
Topic Question Compile Execute Evaluate

```
#include<stdio.h>

void main()
{
int i;

scanf("%d",&i);
printf("%d",i*2);
}
```

CengXXX: Introduction to Lab System

Your programming grade is 100 out of 100.

```
You got 100 out of 100.
```

C Tab Width: 8 Ln 7, Col 16 INS

Conclusions and Important Tips

This laboratory system is designed to make on-computer examinations. So do not rely on the idea “I will learn and do something even if I attend laboratory without any preliminary work”. You have to study the laboratory topic and make some on-computer experiments before attending the laboratory. So that you can learn something on laboratories. Do not forget that the experimenting in a real environment is the key point in programming language learning.

During the laboratory try to form good test data. When a problem is defined, all possible ranges and cases of input is given. You should write a program to cover all these cases, so try to cover all possibilities in your test input.

Appendix (Common Errors)

Programming languages are formal languages and they have very strict syntax rules. One of the most time consuming task in laboratories is the detection and correction of syntax errors. It is always easier not to make errors instead of collecting lots of tiny errors. Actually people make similar mistakes. So please follow these simple rules for the C programming language:

1. Make sure that you include all header files you need. `#include<stdio.h>` line is required for input/output functions and `#include<math.h>` line is required for all mathematical functions.

2. Make sure that all executable statements are separated by a semi-colon ';' even if they are in different lines.

3. Close all matching parenthesis, brackets and curly braces '( ), [ ], { }'. They all should have corresponding matches and they should be properly nested.

4. Do not forget to start and terminate all string literals with a double quote ("). Single quote is just for `char` constants and encloses exactly one symbol. Do not forget to close any such quotes.

5. Do not forget to declare variables that you will be using. All declarations should precede any executable statement. So you will have a declaration part. When all declarations finish the executables should follow.

C is only tolerant to layout of your source. So as long as you follow the syntax rules you can write a C program in a single line. But this is not a good practice. Proper indentation is an important tool for you to understand and develop your problem easily. Write every statement in a separate line and indent inner blocks in the same level.

Although compiler is not tolerant to errors, it provides useful information when reporting the syntax errors. You will see something like:

```
e1036745.c: In function
'main':
e1036745.c:4: Parse error before 'if'
```

This second line reports filename, line number and the exact place and explanation of the error. So this error is at the 4th line and before the word `if`. At the bottom of the development environment you will see which number your cursor is on. So you can easily go to the position of your error.

The following is a list of common errors and how they can be detected-corrected.

1. C program is:

   ```
   #include<stdio.h>
   int main() {
     scanf("%d", &x);
     printf("%d",x*2);
     return 0;
   }
   ```

   You will get this error:

   ```
   te.c: In function `main':
   te.c:3: `x' undeclared (first use in this function)
   te.c:3: (Each undeclared identifier is reported only once
   ```

   You have used a variable called `x` but have not declared it. So insert a line:
   `int x;`
   at the 3rd line position, so that `x` is declared and you can use it.

2. C program is:

   ```
   #include<stdio.h>

   int x;
   scanf("%d", &x);
   printf("%d",x*2);
   return 0;
   ```

   You will get the following error:

   ```
   te.c:3: parse error before string constant
   te.c:3: warning: data definition has no type or storage class
   te.c:4: parse error before string constant
   te.c:4: warning: data definition has no type or storage class
   ```

   In this program a `main()` function is not defined. Compiler expects to have your declarations out of the `main()` function but there are executable statement. So you have to put a `int main() {` line as the 2nd line and add a closing `}` at the end of the program.

3. C program is:

   ```
   #include<stdio.h>
   int main() {
     int x;

     scanf("%d, &x);
     printf("%>d",x*2);
     return 0;
   }
   ```

   You will get the following error:

   ```
   te.c:5: unterminated string or character constant
   te.c:4: possible real start of unterminated constant
   ```

   In this program second quote symbol inside of the `scanf()` at line 4 is forgotten. So C considers all text up to the next quote symbol (5th line, after `printf`) as the first parameter of `scanf`() and than realizes that something is wrong. Try to find the opening quote position and close it in its appropriate place.

4. C program is:

   ```
   #include<stdio.h>
   int main() {
   ```

```
  int x;
  scanf("%d",&x);
  if (x<0) {
    printf("%d",x*2);
  }
  else {
    printf("%d",x+2);
 return 0;
}
```

You will get:

```
te.c:11: parse error at end of input
```

That is because you forget to close the curly brace block opened at line 7 (after `else`). Compiler considers all lines until the last line as the part of the `else` block and closes the `else` block with the `}` at the last line. Then it cannot match the `{` at the 2nd line (`int main() {` ..). Close the curly brace in its appropriate position (after line 8).

5. C program is:

```
#include<stdio.h>
int main() {
  int x;
  scanf("%d", &x);
  if (x<0) {
    printf("%d",x*2);
  }
  else
    printf("%d",x+2);
  }
  return 0;
}
```

You will get:

```
te.c:10: parse error before `return'
```

If you look at the 10th line, before the `return`. You will see the `}` as the preceding item. There is no corresponding `{` for this curly brace. So you should remove it or find the corresponding curly brace position and put it.

6. C program is:

```
#include<stdio.h>
int main() {
  int x;
  scanf("%d", &x);
  if x<0 {
    printf("%d",x*2);
  }
  else {
      printf("%d",x+2);
   }
 return 0;
}
```

You will get:

```
te.c:5: parse error before `x'
```

At line 5, before `x` there is an `if` statement. Parenthesis should enclose the expression after the `if` statement. So you should convert it to `if (x<0) {`.

7. C program is:

```
#include<stdio.h>
int main() {
  int x;
  scanf("%d", x);
  if (x<0) {
    printf("%d",x*2);
  } else {
    printf("%d",x+2);
  }
  return 0;
}
```

You do not get any syntax error. But your program behaves strangely or reports a `segmentation fault` when you run it. You have forgotten the `&` symbol preceding `x` in the 4th line. Compiler cannot detect this error but at run time `scanf` fails to input the value of `x` (if not segmentation fault it puts your value in a strange place in the memory, not in `x`). So it is a worse error than a syntax error. Be careful in input of primitive values.

8. C program is:

```
#include<stdio.h>
int main() {
  int x;
  scanf('%d',&x);
  printf('%d',x*2);
 return 0;
}
```

You will get:

```
te.c:4: warning: multi-character character constant
te.c:4: warning: passing arg 1 of `scanf' makes pointer from integer without a cast
```

Because you have used single quote instead of double quotes. Although your program succeed to compile, these warnings are important and your program will not work properly. So **take warnings seriously**.