1.1 zyLab training: Basics

While the zyLab platform can be used without training, a bit of training may help some students avoid common issues.

The assignment is to get an integer from input, and output that integer squared, ending with newline. (Note: This assignment is configured to have students programming directly in the zyBook. Instructors may instead require students to upload a file). Below is a program that's been nearly completed for you.

1. Click "Run program". The output is wrong. Sometimes a program lacking input will produce wrong output (as in this case), or no output. Remember to always pre-enter needed input.
2. Type 2 in the input box, then click "Run program", and note the output is 4.
3. Type 3 in the input box instead, run, and note the output is 6.

When students are done developing their program, they can submit the program for automated grading.

1. Click the "Submit mode" tab
2. Click "Submit for grading".
3. The first test case failed (as did all test cases, but focus on the first test case first). The highlighted arrow symbol means an ending newline was expected but is missing from your program's output.

Matching output exactly, even whitespace, is often required. Change the program to output an ending newline.

1. Click on "Develop mode", and change the output statement to output a newline: `printf("%d", userNumSquared);`. Type 2 in the input box and run.
2. Click on "Submit mode", click "Submit for grading", and observe that now the first test case passes and 1 point was earned.

The last two test cases failed, due to a bug, yielding only 1 of 3 possible points. Fix that bug.

1. Click on "Develop mode", change the program to use * rather than +, and try running with input 2 (output is 4) and 3 (output is now 9, not 6 as before).
2. Click on "Submit mode" again, and click "Submit for grading". Observe that all test cases are passed, and you've earned 3 of 3 points.

LAB ACTIVITY | 1.1.1: zyLab training: Basics | 3 / 3

main.c | Load default template...

```c
#include <stdio.h>

int main(void) {
   int userNum;
   int userNumSquared;


   scanf("%d", &userNum);

   userNumSquared = userNum * userNum;    // Bug here; fix it when instructed

   printf("%d\n", userNumSquared);          //Output formatting issue here; fix it whe

   return 0;
}

```


Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

2

Run program Input (from above) → **main.c** (Your program) → Output (shown below)

Program output displayed here

1.2 zyLab training: Interleaved input / output

Auto-graded programming assignments have numerous advantages, but have some challenges too. Students commonly struggle with realizing that example input / output provided in an assignment's specification interleaves input and output, but the program *should only output the output parts*. If a program should double its input, an instructor might provide this example:

```
Enter x:
5
x doubled is: 10
```

Students often incorrectly create a program that outputs the 5. Instead, the program should only output the output parts:

```
Enter x:
x doubled is: 10
```


The instructor's example is showing both the output of the program, AND the user's input to that program, assuming the program is developed in an environment where a user is interacting with a program. But the program itself doesn't output the 5 (or the newline following the 5, which occurs when the user types 5 and presses enter).

Also, if the instructor configured the test cases to observe whitespace, then according to the above example, the program should output a newline after **`Enter x:`** (and possibly after the 10, if the instructor's test case expects that).

The program below *incorrectly* echoes the user's input to the output.

1. Try submitting it for grading (click "Submit mode", then "Submit for grading"). Notice that the test cases fail. The first test case's highlighting indicates that output 3 and newline were not expected. In the second test case, the -5 and newline were not expected.
2. Remove the code that echoes the user's input back to the output, and submit again. Now the test cases should all pass.

LAB ACTIVITY | 1.2.1: zyLab training: Interleaved input / output | 2 / 2

main.c — Load default template...

```
1  #include <stdio.h>
2
3  int main(void) {
4     int x;
5
6     printf("Enter x: \n");
7     scanf("%d", &x);
8
9     //printf("%d\n", x);  // Student mistakenly is echo'ing the input to output to matc
10    printf("x doubled is: %d\n", 2 * x);
11
12    return 0;
13 }
14
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

5

Run program | Input (from above) → main.c (Your program) → Output (shown below)

Program output displayed here

1.3 LAB: Formatted output: Hello World!

Write a program that outputs "Hello World!" For ALL labs, end with newline (unless otherwise stated).

LAB ACTIVITY | 1.3.1: LAB: Formatted output: Hello World! 10 / 10

main.c Load default template...

```
#include <stdio.h>

int main(void) {

   /* Type your code here. */
   printf("Hello World!\n");
   return 0;
}

```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program | Input (from above) → main.c (Your program) → Output (shown below)

Program output displayed here

1.4 LAB: Formatted output: No parking sign

Write a program that prints a formatted "No parking" sign as shown below. Note the first line has two leading spaces. For ALL labs, end with newline (unless otherwise stated).

```
  NO PARKING
2:00 - 6:00 a.m.
```

LAB ACTIVITY | 1.4.1: LAB: Formatted output: No parking sign 10 / 10

main.c Load default template...

```
1 #include <stdio.h>
2
3 int main(void) {
4    /* Type your code here. */
5    printf("  NO PARKING\n");
```

```
    printf("2:00 - 6:00 a.m.\n");
    return 0;
}
```


Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program → Input (from above) → **main.c** (Your program) → Output (shown below)

Program output displayed here

1.5 LAB: Input and formatted output: Right-facing arrow

Given two input integers for an arrow body and arrowhead (respectively), print a right-facing arrow.

Ex: If the input is:

```
0 1
```

the output is:

```
    1
    11
0000111
00001111
0000111
    11
    1
```


LAB ACTIVITY | 1.5.1: LAB: Input and formatted output: Right-facing arrow | 10 / 10

main.c — Load default template...

```c
#include <stdio.h>

int main(void) {
   int baseInt;
   int headInt;

   /* Type your code here. */
   scanf("%d/n", &baseInt);
   scanf("%d/n", &headInt);
   printf("    %d\n", headInt);
   printf("    %d%d\n", headInt, headInt);
   printf("%d%d%d%d%d%d%d\n", baseInt, baseInt, baseInt, baseInt, headInt, headInt, he
   printf("%d%d%d%d%d%d%d%d\n", baseInt, baseInt, baseInt, baseInt, headInt, headInt,
   printf("%d%d%d%d%d%d%d\n", baseInt, baseInt, baseInt, baseInt, headInt, headInt, he
   printf("    %d%d\n", headInt, headInt);
   printf("    %d\n", headInt);
   return 0;
}
```

Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

0 1

Run program Input (from above) → **main.c** (Your program) → Output (shown below)

Program output displayed here

1.6 LAB: Warm up: Hello world

This zyLab activity prepares a student for a full programming assignment. Warm up exercises are typically simpler and worth fewer points than a full programming assignment, and are well-suited for an in-person scheduled lab meeting or as self-practice.

For each of the following steps, end the program's output with a newline.

(1) Write a program that outputs the following. (1 pt)

```
Hello world!
```

(2) Update to output the following. (1 pt)

```
Hello world!
How are you?
```

(3) Finally, update to output the following. (1 pt)

```
Hello world!
How are you?
   (I'm fine).
```

LAB ACTIVITY 1.6.1: LAB: Warm up: Hello world 3 / 3

main.c Load default template...

```
#include <stdio.h>

int main(void) {
   /* Type your code here. */
   printf("Hello world!\n");
   printf("How are you?\n");
   printf("   (I'm fine).\n");
```

```
8    return 0;
9 }
```


Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.c** (Your program) → Output (shown below)

Program output displayed here

1.7 LAB: Warm up: Basic output with variables

This zyLab activity prepares a student for a full programming assignment. Warm up exercises are typically simpler and worth fewer points than a full programming assignment, and are well-suited for an in-person scheduled lab meeting or as self-practice.

A variable like userNum can store a value like an integer. Extend the given program as indicated.

1. Output the user's input. (2 pts)
2. Output the input squared and cubed. *Hint: Compute squared as userNum * userNum.* (2 pts)

3. Get a second user input into userNum2, and output the sum and product. (1 pt)

Note: This zyLab outputs a newline after each user-input prompt. For convenience in the examples below, the user's input value is shown on the next line, but such values don't actually appear as output when the program runs.

```
Enter integer:
4
You entered: 4
4 squared is 16
And 4 cubed is 64!!
Enter another integer:
5
4 + 5 is 9
4 * 5 is 20
```


LAB ACTIVITY 1.7.1: LAB: Warm up: Basic output with variables 5 / 5

main.c Load default template...

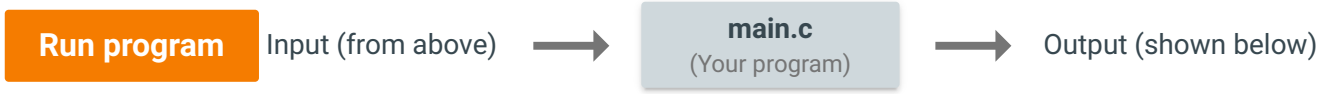
```
1  #include <stdio.h>
2
3  int main(void) {
4     int userNum;
5
6     printf("Enter integer:\n");
7     scanf("%d", &userNum);
8     /* Type your code here. */
9     //printf("%d\n", userNum);
10    int userNumSquared = userNum * userNum;
11    int userNumCubed = userNumSquared * userNum;
12    printf("You entered: %d\n", userNum);
13    printf("%d squared is %d\n", userNum, userNumSquared);
14    printf("And %d cubed is %d!!\n", userNum, userNumCubed);
15    int userNum2;
16    printf("Enter another integer:\n");
17    scanf("%d", &userNum2);
18    int sum = userNum + userNum2;
```


Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

```
4 5
```



Program output displayed here

1.8 LAB*: Program: ASCII art

This zyLab activity is the traditional programming assignment, typically requiring a few hours over a week. The previous sections provide warm up exercises intended to help a student prepare for this programming assignment.

(1) Output this tree. (2 pts)

```
   *
  ***
 *****
*******
  ***
```

(2) Below the tree (with two blank lines), output this cat. (3 pts)

```
/\    /\
  o  o
 =    =
  ---
```


Hint: A backslash `\` *in a string acts as an escape character, such as with a newline* `\n`*. So, to print an actual backslash, escape that backslash by prepending another backslash. Ex: The following prints a single backslash:* `printf("\\");`

LAB ACTIVITY 1.8.1: LAB*: Program: ASCII art 5 / 5

main.c Load default template...

```c
#include <stdio.h>

int main(void) {
   // Draw tree
   printf("   *\n");
   printf("  ***\n");
   /* Type your code here. */
   printf(" *****\n");
   printf("*******\n");
   printf("  ***\n");
   printf("\n\n");
   printf("/\\   /\\\n");
   printf("  o o\n");
   printf(" =   =\n");
   printf("  ---\n");
   return 0;
}
```


Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.c** (Your program) → Output (shown below)

Program output displayed here

©zyBooks 06/03/20 11:56 700057
Rabin Gurung
SJSUCS149SE149AndreopoulosSummer2020