Programming in C I, Group B
Jacobs University Bremen
Dr. Kinga Lipskoch
Xu He

Course: JTSK-320111
September 20th, 2018

Assignment 3 - Conditions and Loops

- The problems of this assignment must be solved in C.
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/320111/2018_2gB/Grading-Criteria-C.pdf

Problem 3.1 *Infinite loop by bad coding* (1 point)

Presence assignment, due by 18:30 h today **Graded manually**

The program below prints

```
i is 8
i is 8
...
```

until you stop the execution by pressing `Ctrl-C`. Fix the program such that it prints 8, 7, 6, 5 and 4 as values for `i`.

```
#include <stdio.h>
int main()
{
    int i = 8;
    while (i >= 4)
        printf("i is %d\n", i);
        i--;
    printf("That's it.\n");
    return 0;
}
```

Problem 3.2 *Divisible by 2 and 7?* (1 point)

Presence assignment, due by 18:30 h today **Graded automatically with testcases only**

Write a program, where you can enter an integer from the keyboard. Determine whether the number is divisible by both 2 and 7. Then either print on the screen
"`The number is divisible by 2 and 7`" or
"`The number is NOT divisible by 2 and 7`".
You can safely assume that the input will be valid.
Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 3.2: input	**Testcase 3.2: output**
`56`	`The number is divisible by 2 and 7`

Problem 3.3 *Categorization of characters* (1 point)

Presence assignment, due by 18:30 h today **Graded automatically with testcases only**

Write a program where you can enter a character from the keyboard. Then determine whether the character is a digit or a letter or some other symbol and print a corresponding message on the screen.
You can safely assume that the input will be valid.
Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 3.3: input

```
!
```

Testcase 3.3: output

```
! is some other symbol
```

Problem 3.4 *Days and hours I* (2 points)

Due by Tuesday, September 25th, 10:00 h **Graded manually**

Write a program where you can enter an integer `n` from the keyboard. Then a conversion table for 1 to `n` days should be printed on the screen as in the example below. Make sure that the integer value you entered for `n` is valid (positive and non-zero). If an invalid integer `n` was entered then repeat the entering until a valid value will be entered.
Use a `while` loop in your solution.

```
1 day = 24 hours
2 days = 48 hours
3 days = 72 hours
...
```

Problem 3.5 *Days and hours II* (1 point)

Due by Tuesday, September 25th, 10:00 h **Graded automatically with testcases only**

In your solution for **Problem 3.4** replace the `while` loop by a `for` loop such that the program will have the same functionality.
Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 3.5: input

```
-8
0
4
```

Testcase 3.5: output

```
Input is invalid, reenter value
Input is invalid, reenter value
1 day = 24 hours
2 days = 48 hours
3 days = 72 hours
4 days = 96 hours
```

Problem 3.6 *Writing numbers* (1 point)

Due by Tuesday, September 25th, 10:00 h **Graded automatically with testcases only**

Write a program where you first enter a float `x` and then an integer `n` from the keyboard. Print the float `x` `n` times to the screen. Make sure that `n` will have a valid integer value. In the invalid case repeat entering `n` until a valid value will be entered.
Your solution has to satisfy the requirements from the problem description and has to pass the following testcase and potentially other testcases which are uploaded. All characters are relevant for passing testcases including newlines and spaces.

Testcase 3.6: input

```
1.25
-8
0
4
```

Testcase 3.6: output

```
Input is invalid, reenter value
Input is invalid, reenter value
1.250000
1.250000
1.250000
1.250000
```

Problem 3.7 *Writing characters I* (1 point)

Due by Tuesday, September 25th, 10:00 h **Graded manually**

Write a program where you first enter a lowercase character `ch` and then an integer `n` from the keyboard. Print the characters `ch`, `ch−1`, ..., `ch−n` on the screen.
You can safely assume that the input is valid and you do not have to do any checks.

Bonus Problem 3.8 *Writing characters II* (1 point)

Due by Tuesday, September 25th, 10:00 h **Graded manually**

Add checks to your solution for **Problem 3.7** such that the program prints corresponding messages and stops (by `return 1` in the `main` function) if the letter is not a lowercase alphabetic character or if `n` is greater than 32 or less than 7.

Problem 3.9 *Computing the average of temperature values* (2 points)

Due by Tuesday, September 25th, 10:00 h **Graded manually**

Write a program where you first enter a character `c` followed by an integer `n` and `n` double values representing temperatures in Celsius. Use an array for storing the temperatures. You can assume that not more than 100 temperature values would be introduced. Your program should compute and/or print the following: if `c` is '`s`' the sum of the temperatures, if `c` is '`p`' the list of all temperatures, if `c` it '`t`' the list of all temperatures in Fahrenheit and if another character was introduced then the arithmetic mean (or average) of all temperatures. The formula for converting Celsius to Fahrenheit is the following:

$$F = \frac{9}{5} \cdot C + 32.$$

Use a `switch` instruction in your solution.
You can safely assume that the input will be valid.

How to submit your solutions

- Your source code should be properly indented and compile with `gcc` without any warnings (You can use `gcc -Wall -o program program.c`). Insert suitable comments (not on every line ...) to explain what your program does.
- Name the programs according to the suggested filenames (they should match the description of the problem) in Grader.
  Each program **must** include a comment on the top like the following:

```
/*
   JTSK-320111
   a3_p1.c
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://grader.eecs.jacobs-university.de`.
  If there are problems (but **only** then) you can submit the programs by sending mail to
  `x.he@jacobs-university.de` **with a subject line that begins with JTSK-320111**.
  It is important that you do begin your subject with the course number, otherwise I might have problems to identify your submission.
- Note, that after the deadline it will not be possible to submit any solutions. It is useless to send late solutions by mail, because they will not be accepted.

This assignment is due by Tuesday, September 25th, 10:00 h.