Assessment 1 – C programming

Demonstration + Online submission; week 9; 10% - group submission

Feedback: within 15 working days

Students will be working in pairs to write code related to the C features they have learned in lab sessions and presenting and submitting it online. The final version of their software will be demonstrated in week 9.

Students will have to develop a C program based on a given criteria. There are 2 possible topics listed below to choose from:

- Dice rolling program
- Easy-to-remember password generator

Students will have to write the code up to good programming standards (correct naming, indentation, comments in code), present it to lab tutors in week 8 lab session and submit it online by the end of week 8.

Students are supposed to use parts of C programming learned in lab sessions (loops, if-statements, variables, functions), but they are encouraged to use any new technologies that were not explicitly taught in lab sessions.

Final assessment will be done in lab session of week 8, students will be supposed to bring the code to the lab session (mail, OneDrive or memory stick), compile it during the presentation, run the code to show it's functional and explain features of the code. Students will be given exact timeslots during the session to be able to accommodate high volume of students. No presence in a given time slot will result in fail mark.

To summarise, project will be marked based on the code functionality, criteria fulfilment as well as code understanding in according to rubric given below.

Dice rolling / Password generator

No submission will result in a Fail mark

Item	1-4	5-8	9-12	13-16	Fail (17-20)
Quality of code 15%	Clean code, using of proper naming standards, code fully commented	Code written up to given standards, high amount of comments	Good structure of code with some mistakes, fair amount of comments	Poor quality of code, minimal number of comments	Poor quality of code, no comments
Technologies used 30%	Use of most required elements (if/else statements, for loops, randomizing values), using technologies that were not used in the lab	Use of most required elements (if/else statements, for loops, randomizing values), no new technologies used	Use of some of required elements(if/else statements, for loops, randomizing values)	Use just one of the required elements(if/else statements, for loops, randomizing values)	Use none of the required elements(if/else statements, for loops, randomizing values)
Fulfilling the task 30%	Achieved or exceeded required functionality, code is running and compiling without errors	Achieved required functionality, code is running some compilation errors are allowed	Achieved most of the required features, code is running some compilation errors are allowed	Achieved part of the required features, code might not compile	Didn’t achieve required functionality, code is not compiling
Understanding of the code 25%	Fully understands the code, able to explain it line by line	Good understanding of code, able to explain most of it	Average understanding of code, able to explain at least half of it	Poor understanding of code, not able to explain more than 25% of code	Lack of understanding of code, not able to explain simple parts of it

Technical Requirements

Topic 1: Dice rolling

Students are supposed to develop a program that will be generating random numbers based on the user inputs. Program is supposed to ask user for a number of faces of a dice and a number of throws (as per example below).

```
stdout      -> Number of faces:

stdin       -> 6

stdout      -> Number of throws:

stdin       -> 10
```

Where stdout is an output of the program and stdin is input of the user. Program is supposed to generate random numbers between 1 and a number of faces and simulate throws based on number of throws given by the user. Program is supposed to store the number of occurrences of each number and calculate percentage of occurrence of each number (see example below).

```
stdout      -> Number of faces:
stdin       -> 6
stdout      -> Number of throws:
stdin       -> 6
stdout      -> Generating throws:
stdout      -> 1
stdout      -> 5
stdout      -> 4
stdout      -> 5
stdout      -> 6
stdout      -> 6
stdout      -> Occurrences of 1: 16.66%
stdout      -> Occurrences of 2: 0.00%
stdout      -> Occurrences of 3: 0.00%
stdout      -> Occurrences of 4: 16.66%
stdout      -> Occurrences of 5: 33.33%
stdout      -> Occurrences of 6: 33.33%
```

Program has to check if the parameters specified by the user are within given ranges, which are as follows:

Number of faces – x, where $1 < x < 25$

Number of throws – x, where $1 < x < 500$

If the parameters given by the user are outside of the range, then program should print a message informing user about invalid parameters and ask the user for parameters once again.

Students will be given a set of functions to generate random numbers but may not choose them in order to achieve first class marks.

Topic 2: Easy-to-remember password generator

Students are supposed to develop a program that will be generating easy- to-remember passwords based on the user inputs. Program is supposed to ask user for a number of words that should be used in the password (as per example below).

```
stdout        -> Number of words:
stdin         -> 3
```

Where stdout is an output of the program and stdin is input of the user. Program is supposed to generate an easy-to-remember password consisting of number of words given by the user. Program is supposed to utilise four 500-word dictionaries (4, 5, 6 and 7 letter long) – dictionaries will be provided by tutors.

Program is supposed to use a random word from a random dictionary and display it to user at each iteration and present a full password to the user in the end of execution (each of the words is supposed to be separated with a hyphen, see example below).

```
stdout        -> Number of words:
stdin         -> 3

stdout        -> Generating password
stdout        -> 1 pretty
stdout        -> 2 dice
stdout        -> 3 tailors
stdout        -> Your e-t-r password is: pretty-dice-tailors
```

Program has to check if the parameter specified by the user is within given ranges, which is as follows:

Number of words – x, where $2 < x < 5$

If the parameters given by the user is outside of the range, then program should print a message informing user about an invalid parameter and ask the user for parameter once again.

Students will be given a set of functions to generate random numbers and iterate through dictionary files but may not choose them in order to achieve first class marks.