**1.** What is the purpose of working on the program after the first version works? Give a list of reasons.

1. Add new features

2. Review the code layout and improve it

3. Test the program one more time and fix remaining bags

**2.** Why does 1+2; q typed into the calculator not quit after it receives an error?

Because q here is considered as a variable, and after the wrong variable name was entered the program won’t quit.

**3.** Why did we choose to make a constant character called number?

In order to avoid magic constant we decided to create the constant, that holds a digit, which indicates the number token. That way it is easier to change it, if we will have to do this

**4.** We split main() into two separate functions. What does the new function do and why did we split main()?

Every function must identify the separate part of code and perform only one logical action. That’s why we decided create a calculation function, which performs all the computation and leave in main only error-handling code.

**5.** Why do we split code into multiple functions? State principles.

Function represents a logical part of code, and ideally does only one logical action.

**6.** What is the purpose of commenting and how should it be done?

Comments are good when something can’t be expressed in code. By commenting we describe other programmers how functionates our program. Every program must be well-commented, but we shouldn’t write comments when the code is obvious. Also we should avoid being verbose while commenting.

**7.** What does narrow\_cast do?

It checks that a value is suitable for the variable of given type.

**8.** What is the use of symbolic constants?

By using symbolic constants we avoid magic constants. The code becomes more error-prone because the value is assigned only in one place of program.

**9.** Why do we care about code layout?

The more program is readable, the less errors we make

**10.** How do we handle % (remainder) of floating-point numbers?

We can use fmod() from the library <cmath>. It will perform % of floating-point numbers. Also, we can test if both of numbers are int with the help of narrow\_cast, and divide with remainder only if they are ints.

**11.** What does is\_declared() do and how does it work?

It checks whether the variable has been already declared or not. It simply goes through the complete names vector and if there is such a variable it returns true.

**12.** The input representation for let is more than one character. How is it accepted as a single token in the modified code?

In get() we check whether the digit is a letter. Then we get the new character and check if it is a digit or a letter. After that we read the whole word in a string variable and check if it is “let” or no.

**13.** What are the rules for what names can and cannot be in the calculator program?

**14.** Why is it a good idea to build a program incrementally?

1. It is easier to debug it step by step, because there is not so much untested code on each step.

2. It is easier to add new features, once we have solved the main problem.

**15.** When do you start to test?

After finishing the debugging

**16.** When do you retest?

When you have added new features, or changed somethin in the program.

**17.** How do you decide what should be a separate function?

If this part of code represents a single logical action, or if there are several actions that are very logicall similiar we should enclose them in the separate function.

**18.** How do you choose names for variables and functions? List possible reasons.

The name should represent the main purpose of the variable or function, and shouldn’t be verbose.

**19.** Why do you add comments?

To express something, that can’t be described in code. Also comments help other programmers to gain a better understanding of how our program works.

**20.** What should be in comments and what should not?

There shouldn’t be comments, when someone who knows programming language can easily understand the code.

**21.** When do we consider a program finished?

When it was successfully tested, solves the initial problem, is well-commented, has good code layout and provides a reasonable output, if it encountered an error.