**Software Engineering Department**

**Computer Organisation and Programming Course   
final assignment**

**Pocket Calculator application**

**Written by:**

**Barak Daniel 205436959**

**&**

**Gal Avital 308493832**

**28 July 2022**

**Lecturer: Dr Yigal Hoffner**

TABLE OF CONTENTS

TABLE OF CONTENTS 1

1 Pocket Calculator application design 2

1.1 Extra Work Carried Out (extra 25% items) 2

1.2 Major Design/Implementation Decisions 2

1.3 The High-level Algorithms 2

2 The User Guide 2

# Pocket Calculator application design

## Extra Work Carried Out (extra 25% items)

Please put in what you did so I will know what to check for. The options are:

1. 5% for fast multiplication.
2. 5% for enabling Back-Space in input.

## Major Design/Implementation Decisions

* Integers are represented internally using the 2's complement representation
* Our main goal was to stick to the example you gave. We designed the program to be very similar to the results in the example.
* At first, we prepared the application chassis and all the printing functions (not result print).
* After that, we built the "PlusOp" function to calculate Res=X+Y, but a function was required to receive characters and convert them into one whole number.
* So, we built a function that would fulfill all the conditions required to receive the characters and turn them into a proper number (if it is proper)
* Then we built a function that would know how to access the function of the reception and receive Both X and Y with all the required conditions.
* Now that we have the two numbers and after checking that they are really valid numbers we were ready to create a function that would print the result.
* Then we made the "MinusOp" function to calculate Res=X-Y and continued first to "DiviOp"
* There were some issues with "DiviOp" function so we made some flags and the code turned a bit uglier.
* We kind of did a "look alike" code duplication, but it was necessary in some cases like "ResultRem" that is very similar to "Result" but prints the remainder inside the parenthesis, the same goes with "DivRem" and "Div" functions.
* The hardest function to rewrite again and again was the "GetSingedShort" so it will handle all the issues of error digit and make sure we are not losing the Minus if needed, especially when we next enabled the Back-Space. Then added the "MultOp" function for fast multiplication.
* All the "high-level" code is located next to the code inside the notepad document. we also left many comments that explain all you need to know, and not leave you with questions or doubts.

## The High-level Algorithms

All the High-level algorithms we used in this assignment are next to the assembly code in the notepad document. On top of that we also added comments wherever needed.

# The User Guide

1. **Instructions on how to use the application-**

We made the application as similar as possible to your example.

First you get the opening sentence and then you are asked to enter operation sign, no need to press enter after selecting a sign, we found it much more efficient and convenient this way (especially when we are not using sophisticated input).

Now you get echo: “ 1st: ” and the user insert the number, there is no problem to enter the sign Minus. Obviously only in the first digit.

After the user finish entering the digits to a full wanted number just press “Enter (CR)”, user will get echo: “ 2nd : ” then similar to the first number enter the digits to a full number and press “Enter (CR)”.

User will now get a result printed after the sign “equal” =.

If the user entered a wrong digit, at any stage, he will receive an error message (Even if he doesn’t press enter( and restart from selecting an operation without terminating the entire application. OFC we will not print the long sentence again… same goes for “division by 0” and if the operation sign inserted from the user is incorrect.

If the user has entered a digit that is **valid** for the step he is in, he can always press backspace and continue the calculation from that point. You can only backspace 1 digit each time don’t try to backspace twice in a row to delete two digits at once. OFC you can insert digit then press backspace 1 digit then insert another digit and press backspace again no problem.

1. **Extras** - We chose fast multiplication and we enable Back-Space in input TOTAL OF 10%.
2. **Anything else you think is important for the user like limitations or problems with your application. -** To be honest, if we had to reprogram the calculator it seems we would do it differently, but it's easy to say now. When we started the task and slowly progressed step by step, we could not always think of all the problems that could arise. Accordingly, we sometimes had to add flags or not the most "pretty" solutions to our code. But looking back and according to our knowledge at the beginning of the task, and it is important to note that this is the first big task we are doing at Assembly, we are very satisfied with the final result. We believe that the code is unbreakable according to the mission instructions. Perhaps it would have been possible to plan differently and make the code “cleaner”. There are few things that could be improve, especially in planning and design, but despite everything, the bottom line is that the code is unbreakable and does its job well.