CM10227/50258 Principles of Programming
Coursework 1: SRPN

1 Introduction

This document provides the specification for the first Principles of Programming coursework.

You can use any environment for the development of your script, but we must be able to compile and run your code using **Python 3.8** without requiring the installation of additional libraries, modules or other programs.

Questions regarding the coursework can be posted on the Moodle forums, sent to the programming1@lists.bath.ac.uk mailing list, or asked in labs/lectures.

2 Learning Objectives

At the end of this coursework you will be able to design and write a medium-sized program using the appropriate procedural software techniques of data encapsulation and decomposition. You will also be able to understand, construct and use key data structures in a program.

3 Saturated Reverse Polish Notation (SRPN) Calculator

Whilst performing some maintenance on a legacy system you find that it makes use of a program called **SRPN**. **SRPN** is not documented and no one seems to know who wrote it, so your boss tells you to rewrite it in Python.

SRPN is a reverse polish notation calculator with the extra feature that all arithmetic is saturated i.e. when it reaches the maximum value that can be stored in a variable, it stays at the maximum rather than wrapping around.

Your task is to write a program which matches the functionality of *SRPN* as closely as possible. Note that this includes *not* adding or enhancing existing features.

You have been provided with a repl.it link which allows you to run and interact with the SRPN program. You should spend time inputting data and observing the output to understand what the program does.

On the following pages we have provided four categories of tests:

- Single operations
- Multiple operations
- Saturation
- Obscure functionality

In each category we have provided **example** test data that you should input and observe the output of. A good starting point in this coursework is to write code that meets these tests and variations of the data.

1. Single operations. The program must be able to input at least two numbers and perform one operation correctly and output

```
Input:

10
2
+
=

Input:

11
3
-
=

Input:

9
4
*
=

Input:

11
3
/
=

Input:

11
3
%
=
```

2. Multiple operations. The program must be able to handle multiple numbers and multiple operations

```
Input:

3
3
*
4
4
*
+
=

Input:

1234
2345
3456
d
+
d
+
d
=
```

3. Saturation. The program must be able to correctly handle saturation

```
Input:

2147483647
1
+
=

Input:

-2147483647
1
-
=
20
-
=

Input:

100000
0
-
d
*
=
```

4. Obscure functionality. The program includes the less obvious features of SRPN. These include but are not limited to...

```
Input:

1
+

Input:

10
5
-5
+
/

Input:

11+1+1+d

Input:

# This is a comment #
1 2 + # And so is this #
d

Input:

3 3 ^ 3 ^ 3 ^=

Input:

r r r r r r r r r r r r r r r r r r r r d r r r d
```

3.1 Investigating and getting started with SRPN

This is an investigative assignment. Once you have tried the inputs described above, you should spend some time trying other inputs into the calculator. The legacy SRPN calculator has a lot of functionality, some of it obscure. We are looking for you to recreate the same calculator, *not to fix its flaws or to add new functionality.*

It is important that your outputs are the same as the SRPN calculator. For example, SRPN's "Stack overflow." is not the same as "stack overflown" – note the spelling error, lack of capitalisation and missing full stop.

We will run the above tests, as well as others that are similar or more obscure. You are not expected to find and implement every feature. Start by implementing functionality to meet some of the above tests, adding bits of functionality at a time, continually testing that your solution works before adding more.

It is possible to spend many hours working on this assignment to attempt to replicate it perfectly – do not do this. You should manage your time to implement the core functionality with some additional features if possible, then dedicate time to commenting your code and improving your code quality.

3.1.1 Automated tests

You have been provided with some starting code `srpn.py` and a test script `mark-code.py` that will run the above example inputs. As you develop your code, you should run it against the test script to check that it meets expected outputs.

You can run the test script in repl.it by going to the *Shell* window and entering `python mark-code.py`.

This assignment will be marked against a test script similar to the one provided. It will run additional tests covering the same set of functionality, as well as some more advanced features.

3.1.2 Advanced Features

You have also been provided with a second version of the program, **SRPN-Advanced**. SRPN-Advanced includes some more complex functionality in how it handles inputs. You stand to gain additional marks for replicating this advanced functionality.

If you are new to programming, you are suggested to focus on the standard SRPN.

4 Submission Instructions

Your source (`.py`) files must be submitted in a single zip, with all files in the root of the zip. You must not include subfolders within the zip file. Your zip file name must be of the form `username-srpn.zip`, e.g. `abc12-srpn.zip`.

Your .zip file must be submitted to the Moodle assignment page by the submission deadline shown. Submissions received after this deadline will be capped at 40% if received within 5 working days. Any submissions received after 5 working days will marked at 0%. If you have a valid reason for an extension, you must submit an extension request through your Director of Studies – unit leaders cannot grant extensions.

You should leave yourself time to download your file from Moodle, extract it, and check that you have attached the correct file, with the content that you want to be marked. **You** are responsible for checking that you are submitting the correct material to the correct assignment.

Please check Moodle for the submission deadline.

5 Assessment

5.1 Conditions

The coursework will be conducted individually. Attention is drawn to the University rules on plagiarism. While software reuse (with correct referencing of the source) is permitted, we will only be able to assess your contribution.

5.2 Marking

This code will be marked solely on functionality. You will be awarded marks for your code successfully running with the marking script. You should still dedicate time in this coursework to commenting your code and ensuring your code is well-written, as an ongoing goal of furthering your programming skills. Below is a breakdown of marks, with descriptions on how each criteria is met.

Criteria	Max Score	Description
Running with expected input	max 70	Code replicates the functionality of the SRPN program to read input and produce expected outputs. This includes handling incorrect user inputs.
Advanced features	max 30	Code replicates the functionality of the SRPN-Advanced program.

In cases where it is not clear from the assignment how it should be marked, you may be called on to explain or demonstrate your program. Such cases include but are not limited to suspected plagiarism.