BT Code Test Generic Rules

*Please read these instructions carefully, particularly those under **Submitting your code**.*

We will accept submissions in any of the following languages. Except where otherwise noted, submitted code will be compiled/run under Ubuntu GNU/Linux or Windows 7 using the language versions shown below. You may use a higher version of the language (e.g. Ruby 2.0 instead of Ruby 1.9), but if you do so please indicate in your submission which version you have used.

Language	Compiler/Interpreter Version
C	gcc 4.6
C++	g++ 4.6
C#	See note 1
Java	JDK 1.7. See note 2
JavaScript	Node.js 0.10
Objective-C	See note 3
Perl	5.14
PHP	5.3
Python	2.7
Ruby	1.9
VisualBasic	See note 1

- **Note 1:** VisualBasic and C# are as supported by VisualStudio Express 2013 for Windows Desktop.
- **Note 2:** Java code will be assumed to be platform independent and to compile under JDK 1.7.
- **Note 3:** Objective-C will be tested on Apple Xcode 5 on a recent version of OS X.

These languages represent a selection of those in most widespread use today. All of them are in use within BT, but so are many others – a number of which are more widely used than some of those that appear in the table! If you have a compelling reason to want to use a different language – such as having no experience with any of those listed – then please contact us to determine whether we have the capability to accept such a submission.

Command-line ("console") applications

Unless otherwise stated in the description of the problem, programs must be command-line ("console" or "terminal") applications. Under Windows this means that they will be run from the command shell, `cmd.exe`. Under Linux, MacOS or other UNIX-like systems this means that that will be run from a `bash` shell within a terminal window. This applies even if the more

normal mode of using the language would be with another kind of interface — such as a GUI application with VisualBasic or a web application for JavaScript or PHP.

Unless otherwise stated programs will take a number of command-line argument and write their results to standard output (the console/terminal). So they will be run something like this:

```
MyProgram input.txt foo bar
```

This means that `MyProgram` will be invoked with three command-line arguments, "`input.txt`", "`foo`" and "`bar`".

Submitting your code

Your submitted program code may consist of multiple source files but you must include a prominent "`README.txt`" file that gives instructions for building and running your program. You can also include with your program code any other data files, unit tests and so on to demonstrate that it works – but do not include any pre-compiled files (e.g. `.jar`, `.class`, `.obj`, `.exe`, `.dll` etc.).

You may either send a link to files in a publicly-accessible code repository (e.g. GitHub, BitBucket or Google Code) or e-mail your code as a zipped attachment to an e-mail. If you send your code via e-mail we recommend that you add a "`.txt`" extension to any interpreted language files within the archive — e.g. `MyProgram.py` becomes `MyProgram.py.txt` — because our firewalls will block files (including those in zip archives) that appear to be executable scripts. These files include JavaScript, Perl, PHP, Python, Ruby and VisualBasic source files as well as compiled files like `.exe` and `.dll`. If in doubt, add a `.txt` suffix!

Other requirements

Your program must not rely on any libraries that do not form part of a normal base installation of the language, but you may use anything from within the language's standard libraries. So for example, unless explicitly asked to do so, we wouldn't expect you to implement your own sort routine if there is already one available in the standard libraries. Please be aware that your program should, where possible, avoid operating system-specific assumptions such line ending characters or file-name path delimiters.

We will test your code, and expect you to have done so prior to submitting it. You will not automatically "fail" if we uncover bugs or limitations in your program. However, please be prepared to be challenged over improvements and possible fixes when we discuss your code with you. We will test your code with input sets other than those supplied in the problem description, so please expect that.

You may find that the specification of the problem seems ambiguous, or not as complete as you would like. That is, of course, part of the reality of software development! If you find that you have to make certain assumptions in order to implement your program then please do

so, but make it clear what those assumptions are. We will not provide you with further examples or test cases – although if you find something in the problem specification that seems contradictory or just plain wrong then please let us know.

It should go without saying that the program must be your own work. You are free to take inspiration and even small code snippets from other sources, but you must acknowledge if you do so with appropriate comments. In any event you will be expected to be able to explain in detail how every part of your program works.