Assignment: Programming Languages Report

Date Due: 4pm, MARCH 23RD 2015

Your report should be printed out, and placed in the box labelled CSC9Y4 outside 4B89. You must also submit your report to TurnItIn via the link on the course Succeed page. Within three working days of collection the course organiser will post a notice on the board outside 4B80 and on the CSC9Y4 website with a list of assignments not received. It is your responsibility to check this list to ensure your assignment has been successfully submitted.

If you cannot meet the assignment hand in deadline and have good cause, please see John Woodward (4B102, jrw@cs.strath.ac.uk) to explain your situation and ask for an extension. University Regulations state that coursework will be accepted up to seven days after the hand in deadline (or expiry of any agreed extension) but the grade will be lowered by three grade marks per calendar day or part thereof. After seven days the work will be deemed a non-submission and will receive no grade. The further consequence of this will be a no grade for the module overall.

Marking Allocation. Marks will be allocated at 30% for presentation of the report and 60% for technical content. The remaining 10% of marks is for demonstrating a higher understanding of writing approach: you should list five writing tips you have applied in your report, with a brief indication of how they apply in your work. See e.g. the tutorials, or as (http://personal.strath.ac.uk/d.j.higham/tips.html and http://personal.strath.ac.uk/d.j.higham/tips.pdf). These only need to be 5 bullet points, but it shows you have applied a technique to your writing.

Write a report on ONE of the topics. These are not topics covered specifically in the course. This is an exercise in researching programming language features, using the knowledge you have of programming languages and paradigms in general, and then writing coherently about them.

You are encouraged to make use of several references in the writing of your report. Extra credit is given to reports using references other than those listed below. The given URLs should therefore be regarded as the starting point for an information search. These will be available on the course website for convenience. For computing topics, Wikipedia is generally a reliable source; however, the best reports draw their information from a wide variety of sources.

However, you should take care to express what you have learned in your own words. Plagiarism is not acceptable. Work submitted for assessment must be your own work. All students should note that the University has a formal policy on plagiarism which can be found at www.quality.stir.ac.uk/ac-policy/assessment.php

See also our new departmental guidance for students at www.cs.strath.ac.uk/guides/avoiding-plagiarism.pdf. In particular, it is not permitted to take large portions of someone else’s work and use these directly in your assignment. If you directly quote from an article, make this clear by using inverted commas or putting the quote in an indented paragraph on its own. You must also give a reference to the source of the quote in the form (Wilson and Clark, 2000). For web pages use the title if no author is credited. If used, such quotations should form a small part of the report. A list of all references must be given at the end of the report in the bibliography. A useful resource for formatting references and the bibliography may be found at www.arereasearchguide.com/9parenth.html.

All submissions must be checked for originality using the TurnitinUK plagiarism detection software. This software checks the content of your report for similarities with billions of
documents on the Internet, journal articles, and student submissions. A poor report from the
software will lead to further investigation of plagiarism in your report. You may submit your
report to the TurnitinUK software repeatedly before the assignment is due to ensure
originality of expression.

Your report should be structured around the position presented, with which you may either
agree or disagree. Either way, a good report will be composed of sections with suitable
headings such as Introduction (stating the proposition and including context for your report),
one or more technical sections (giving the background of the scenario in more detail,
relevant parts of programming language design for your argument, points for and against the
position, criteria for judging the argument), Conclusions (are you for or against?), and any
other sections necessary. Where possible, include diagrams and simple example programs
to clarify your text. Reports should be about 7-10 pages (e.g. in font size 11pt). This is
roughly equivalent to 3000-4000 words. Very long reports will not be viewed favourably
unless of exceptional quality. Remember Quality not Quantity. This assignment counts for
25% of the final mark in CSC9Y4. It must be submitted in order for you to be given a grade
for the course.

1. **Mobile users and the Cloud**

but now buy them and own them outright. Housing and automobiles have also witnessed
similar migrations of ownership. Is computing any different? With innovations in cloud
computing, such as dropbox and Google drive/docs, we are seeing a reversal in this trend.

In the past, computer users sat in front of dumb terminals (i.e. literally just a keyboard and a
monitor) and accessed applications and data on a remote server. Often users had no
knowledge of the precise capabilities or location of that server. Having been through a phase
of desktop computing, where all resources are in the computer in front of you, we are
returning to the situation where resources are remote and the computer in front of you has
rather limited computation power and memory. Scripting approaches are increasingly
common as users plug together components available remotely.

What are the implications for programming language design and use of returning to a more
lightweight application approach? Do you agree with the position above?

**References**

d1smfj0g31qzek.cloudfront.net/abovetheclouds.pdf
cloudcomputing.sys-con.com
en.wikipedia.org/wiki/MapReduce
boinc.berkeley.edu
scg.unibe.ch/research/piccolo
searchitchannel.techtarget.com/feature/Cloud-applications-A-programming-languages-background

2. **Dynamic Typing is great for beginners**

Position: Being freed from excessive syntax, especially declarations of variables, means
languages such as Python allow beginner programmers to learn more quickly.
Python is a dynamically-typed strongly-typed object-oriented programming language, with a steadily growing community of users. Several institutions use Python as a first teaching language. Identify Python’s advantages for teaching, and compare these with your experience in Java. Concentrate specifically on features which might appeal to someone learning programming for the first time, rather than an experienced programmer converting to a new language. Does dynamic typing make programming easier? Safer? Is coding easier to learn? Do you get simpler programs?

References

www.python.org
en.wikipedia.org/wiki/Dynamic_typing
mcsp.wartburg.edu/zelle/python/python-first.html

3. Natural Language Processing

Position: Speech recognition has now advanced to the point where all users should throw away their keyboards and only communicate by voice with their computers.

Typical science fiction includes computers communicating with people in a natural way, with spoken input and output, mimicking ordinary speech. The advent of sophisticated voice command applications (e.g. Siri, Dragon NaturallySpeaking) suggest this sci-fi future is a reality.

Have we really reached the pinnacle of natural language processing, or is there much more work to do? What are the problems remaining to be solved? Why are natural languages so much harder to process than computer languages? What sort of programming languages can we use to process natural languages? Traditionally, languages such as Lisp and Prolog have been used, but scripting languages are also popular.

References

research.microsoft.com/nlp/
en.wikipedia.org/wiki/Natural_language_processing
www.aaai.org/AITopics/html/natlang.html
www.alicebot.org/

Natural Language processing is in section SC 22.7 of the library, with speech processing at 24.5.

4. Programming for All

Position: The ability to program should become as widespread as the ability to read, swim or ride a bike.

We all own computers or computational devices such as mobile phones or tables. Also there are more teaching resources available, commonly known as MOOCS. Can anyone learn to program? Can children learn to program? What features should a programming language
have to make it easy to learn (think about the programming languages you have been exposed to over this course). Are there some concepts which are just too difficult for the majority of people to learn such as higher order functions and partial functions associated with functional programming? How far can we go, certainly most people can use a pocket calculator without too much difficulty. Will programming make the leap from being a pursuit of the privileged few to the general public?

References
http://learnyouahaskell.com/
http://en.wikipedia.org/wiki/Massive_open_online_course
https://www.coursera.org/
http://barclayscodeplayground.co.uk/

5.  Visual Programming

Position: Now that touch screen devices are more ubiquitous, it will soon be routine to construct programs visually.

Programs normally consist of ASCII text but for a long time, programmers have tried to develop more visual approaches to programming, often focusing on languages consisting of diagrams rather than text. For example, Alice from Carnegie Mellon University allows students to program a 3D world using drag and drop. Blockly is another similar project.

Outline the main features of visual programming languages and discuss their advantages and disadvantages for the programmer. What is the purpose of making a language visual? Is it possible to write large and complex programs in this way? Or is visual programming just for novices and therefore a bit of a fad?

References
blog.interfacevision.com/design/design-visual-programming-languages-snapshots
en.wikipedia.org/wiki/Visual_programming_language
www.alice.org
code.google.com/p/blockly
www.tersus.com

6.  Big Data Programming

Position: Handling big data is the same as handling small data: there's just more of it.

Data Science is a fast growing area of computing. We now have large numbers of sensors producing vast amounts of data all the time. We also create much of our own data via social media. What do we do with this data? How can we make best use of it?
Outline the main features of big data processing, and discuss special requirements they might have of programming languages. Can you just use traditional programming languages? Is it about component engineering? Or MapReduce? Do we need a new paradigm to talk elegantly and efficiently about big data processing?

References

www.datascientists.net/what-is-data-science
www.ibmbigdatahub.com/blog/data-scientists-myths-and-mathemagical-superpowers
readwrite.com/2013/11/25/python-displacing-r-as-the-programming-language-for-data-science#awesm=oyILpZljeejQBI
en.wikipedia.org/wiki/Programming_with_Big_Data_in_R

7. Parallel Programming

Position: The technological innovation of parallel programming is becoming the norm, while serial processing will become obsolete and fall by the wayside. However, programming languages are already complex, so we should only expose the programmer to the problem of programming a serial machine, and let the compiler deal with the issues of porting serial code to a parallel architecture.

Parallel programming is generally much faster than serial processing. This is particularly true when we want to carry out a heavy computation – we could just rent a huge number of parallel processors in the cloud for a short space of time, rather than being limited by the hardware we own. Parallel programming will become more widespread in the next few years. Therefore we should teach it as early as possible in the undergraduate syllabus. In fact, we should go so far as to stop teaching serial programming, as in a few years, single processor machines will not exist. Indeed one might argue that the human brain itself is massively parallel; made of “slow” neurons (which activate over 10s of milliseconds) compared to billions of calculations carried out serially on modern day computer chips.

References

http://en.wikipedia.org/wiki/Parallel_computing
http://neuralnetworksanddeeplearning.com/chap1.html
https://computing.llnl.gov/tutorials/parallel_comp/

8. Bring your own

You may have your own idea for an essay. This is very welcome as this can motivate some students. Do you have a particular interest in a programming language, or would like to develop you thoughts on a certain aspect of programming you have learnt on the course? Please drop me an email first to clear it with me (jrw@cs.stir.ac.uk).