RHODES UNIVERSITY

Computer Science 301 - 2016 - Programming Language Translation

Well, here you are. Here is the free information you have all been waiting for, with some extra bits of advice:

- Don't panic. It may be easier than you might at first think.
- The problems in the examination based on the exercise posed below will need rather careful thought. I shall be looking for evidence of mature solutions, not crude hacks.
- Work in smaller, rather than larger groups. Too many conflicting ideas might be less helpful than a few carefully thought out ones.
- Do make sure you get a good night's sleep!

This year the format of the compilers examination is similar to that of the last few years. The problem set below is only part of the story. At about 16h30 you will receive further hints as to how this problem should be solved (by then I hope you might have worked this out for yourselves, of course). You will be encouraged to study the problem further, and the hints in detail, because during the examination tomorrow you will be set further questions relating to the system - for example, asked to extend the solution you have worked on in certain ways. It is my hope that if you have really understood the material today, these questions will have solutions that come readily to mind, but of course you will have to answer these on your own.

My "24 hour exam" problems have all been designed so that everyone should be able to produce at least the basic solution, with scope given for top students to demonstrate their understanding of the subtler points of parsing, scanning and compiling. Each year I have been astounded at the quality of some of the solutions received, and I trust that this year will be no exception.

Please note that there will be no obligation to produce a complete working system in the examination (in fact, trying to do so is quite a risky thing, if things go wrong for you, or if you cannot type quickly). The tools will be provided before the examination so that you can experiment in detail. If you feel confident, then you are free to produce a complete working solution to Section B during the examination. If you feel more confident writing out a neat detailed solution or extensive summary of the important parts of the solution, then that is quite acceptable. Many of the best solutions over the last few years have taken that form.

How to spend a Significant Special Sunday

From now until about 22h30 tonight, Computer Science 3 students have exclusive use of the Hamilton Laboratories. You are encouraged to throw out anyone else who tries to use it, but we hope that this does not need to happen. At about 22h30 we may have to rearrange things for tomorrow. If there is still a high demand we shall try to leave some computers for use until later, but by then you should have a good idea of what is involved.

Today you will be able to find the following files in the usual places:

- All the files as were made available for the practicals, tests, solutions, past papers.
- The file `FREE1.ZIP`, which contain the C# versions of the Coco/R system, and its support files, some grammar and skeleton files and test data for today's exercise. The kit also contains a PDF version of the Coco/R user manual (`CocoManual.pdf`) and library summaries (`Library.pdf`).

At about 16h30 a new version of the exam kit will be posted (`FREE2.ZIP`), and a further handout issued, with extra information, to help students who may be straying off course.

So today things should be familiar. You could, for example, log onto the `D:` or `J:` drive, use `NotePad++` and `LPRINT` to edit and print out files, use `CMAKE` to build Parva … generally have hours of fun.

Note that the exam setup tomorrow will have *no* connection with the outside world - no Google, FaceBook, ftp client, telnet client, shared directories - not even a printer.

Today you may use the files and systems in any way that you wish, subject to the following restrictions: *Please observe these in the interests of everyone else in the class.*

(a) When you have finished working, **please** delete your files from any shared drives, so that others are not tempted to come and snoop around to steal ideas from you.

(b) You are permitted to discuss the problem with one another, and with anybody not on the "prohibited" list.

(c) You are also free to consult books in the library. If you cannot find a book that you are looking for, it may well be the case that there is a copy in the Department. Feel free to ask.

(d) Please do not try to write any files onto the C: drive, for example to `C:\TEMP\`

(e) If you take the exam kit to a private machine you will need to have the .NET framework installed.

I suggest that you *do* spend some of the next 24 hours in discussion with one another, and some of the time in actually trying out your ideas. If you have prepared properly for the exam you should have plenty of time in which to implement and test your ideas - go for it, and good luck.

If you cannot unpack the file, or have trouble getting the familiar tools to work (unlikely!), you may ask me for help. You may also ask for explanation of any points in the question that you do not understand, in the same way that you are allowed to ask before the start of an ordinary examination. You are no longer allowed to ask me questions about any other part of the course. Sorry: you had your chance earlier, and I cannot allow this without risking the chance of sneak questions and suggestions being called for.

If you cannot solve the problem completely, don't panic. It has been designed so that I can recognize that students have reached varying degrees of sophistication and understanding.

How you will spend a Merry Mind-blowing Monday Morning

Before the start of the formal examination the laboratory will be unavailable. During that time

- The machines will be completely converted to a fresh exam system with no files left on directories like `D:` or `C:\TEMP` .

- The network connections will be disabled.

At the start of the examination session:

- You will be allocated to a computer and supplied with a `CONNECT` command for your own use. Once connected you will find an exam kit on the `J:` drive. This will contain the same Coco/R system and other files you have been given today, and in addition there will be a "flat ASCII" machine readable examination paper as well as an MS-Word version.

- You will receive an examination paper, with spaced questions so that, if you like, you can write your answers on the exam paper itself.

- You will receive listings of various of the grammars and support files that you receive today. *You may annotate these during the exam to form part of your solution if you wish to submit hand-written answers to questions, and need to make reference to the code (possibly by the line numbers that are provided on the listings).* In this case you should hand in the annotated listings with your exam paper.

- *There is no obligation to use a computer during the exam. You can answer on the examination paper if you prefer - and yes, you can write in pencil if you prefer that - but please not in red ink.*

- At the end of the exam you will be given a chance to copy any files that you have edited or created on the D: or J: drive back to the server. This is done using a simple script and will be explained tomorrow.

- **Remember that tomorrow you may not bring anything into the room other than your student card and writing utensils, and especially not listings, disk drives, memory sticks, text books or cell phones.**

Preliminary to Section B of the examination

It was in 1988, at the second of a series of conferences held by a group charged with developing a rigorous standard for the definition of Modula-2, that one Susan Eisenbach made a telling observation: "Modula-2 would be the perfect language if we could add just one more feature. The difficulty is that nobody can decide what that one feature should be". She was right. Language designers cannot resist the temptation to add extension after extension to their brainchild.

The Parva language that you have got to know so well, currently has three basic data types - integer, boolean, and character. But people expect it to have other familiar features, and are alarmed and annoyed when it is found that some are missing.

Coincidentally, a minor crisis has developed in the Computer Science department. This year, as an experiment, the first year students in our popular CSC-ARB course have been taught programming using the popular Parva language. This has proved very successful, and the students are all expected to pass their practical examination on Monday. Today is Sunday. Earlier this morning the lecturer in charge of the class discovered, to her horror, that the existing Parva compiler makes no provision for the *set* type that would be required if students were asked to write code like the following:

```
// A class tries to guess their lecturer's age
// P.D. Terry,  Rhodes University, 2016

void Main () {
  int guess, total = 0, guesses = 0;
  set uniqueGuesses = set{};                      // create an empty set of guesses

  read("Supply first guess (0 stops) ", guess);
  while (guess > 0) {
    uniqueGuesses.Incl(guess);                    // record this guess - maybe again and again
    total = total + guess;
    ++guesses;
    read("Supply next guess (0 stops) ", guess);
  }
  int average = total / guesses;                  // compute the average in the usual way

  int actualAge;
  read("Supply lecturer's actual age ", actualAge);
  if (average <= actualAge)
    writeLine("You flatter me");
  else
    writeLine("Some DPs will have to be withdrawn");

  if (average in uniqueGuesses)                   // now report on the statistics
    writeLine("At least one student guessed the average of all the guesses");

  if (actualAge in uniqueGuesses)
    writeLine("At least one student guessed the lecturer's age correctly");

  writeLine(Members(uniqueGuesses), " different guesses were made by ",
            guesses, " students");
} // Main
```

They also need to be able to write similar applications which require simple set manipulations, including the ability to create empty sets or sets initialised by a list of values, add elements to an existing set, check for membership of a set, create the union of two sets, write the members of a set and compare and clone sets.

In desperation she turns to the members of the CSC 301 class, imploring them to come up with a reliable new version of the Parva compiler in 24 hours that will provide the *set* type. To test the system, she offers to provide them with a few examples of the sort of code that she expects the compiler to be able to handle.

Rise to the challenge! #ParvaMustNotFall. Produce a new compiler, avoid a disaster for the department and get closer to earning that most prized possession - a degree from Rhodes University.

Where do you go from here?

Take a little time to read through the rest of this document carefully and make sure you understand it before you leap in and start to hack at a "solution".

In adopting the approach suggested below you can (and should) make use of the files provided in the examination kit `free1.zip`, which you will find on the course website. In particular, you will find, after unzipping this, a structure like that below - and it is recommended that you retain this directory structure

```
work\CMake.bat              Script to generate and compile the Parva compiler

work\CocoManual.pdf         Documentation
work\Library.pdf

work\Coco.exe               Coco compiler and basic frame files
work\Scanner.frame
work\Parser.frame
work\Driver.frame
work\Parva.atg              Parva grammar
work\Parva.frame            Parva frame file

work\Library.cs             Rhodes C# library

work\Parva\CodeGen.cs       Auxiliary routines for Parva compiler
work\Parva\PVM.cs
work\Parva\Table.cs

work\examples\eg*.pav       The source code for test programs
```

To allow you to test your compiler on programs that are not meant to be executed (merely compiled so as to check the `.COD` file, syntax or the program listing), remember that the Parva compiler recognizes a `-n` command line flag, as in

```
Parva  voter.pav  -l  -d  -n
```

which won't call on the PVM to interpret the code after it has been generated. It also recognizes a `-g` command line flag, as in

```
Parva  Eg03.pav  -l  -d  -g
```

which will follow a successful compilation with an immediate interpretation without the usual annoying questions.

Hints:

(a) The examination kit includes all the files needed to build a working Parva compiler similar to the one developed in your last practical exercise (it has some, but not all of the extensions you were asked to make in that practical, and you need not bother to try to add the missing extensions again).

(b) It also includes a graded collection of test programs of the sort outlined above (in files named `egXX.pav`). There is an annotated listing of all these programs in a this handout, which also provides a step-by-step guide on how to develop the system.

(c) Depending on your approach, your solution may require modifications to any or all of the grammar and support files in the exam kit.

(d) As with all of the "24 hour exam" problems devised over many years, this is a non-trivial exercise, and you will have to think clearly to be able to solve it.

(e) It is not particularly difficult to provide "first approximations" to these extensions and modifications. The examiners will, however, be looking for maturity in your solutions.

(f) Rest assured that you will not be expected to reproduce a complete Parva compiler from memory under examination conditions, but you may be asked to make some additions or improvements to the system developed today. You will not be asked to add all the missing extensions of the last practical again.

(g) Remember Einstein's Advice: "Keep it as simple as you can but no simpler" and Terry's Corollary: "For every apparently complex programming problem there is an elegant solution waiting to be discovered".