RHODES UNIVERSITY

Computer Science 301 - 2002 - 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 is probably easier than you might at first think.
- The solution to section B does not need an enormous amount of code. But it needs rather careful thought. I am 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!

How to spend a Special Sunday

From now until about 10h30 tonight, Computer Science 3 students have exclusive use of the Hamilton Laboratory. 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 10h30 pm Jody and I will have to move some computers around and prepare things for Monday. If there is still a high demand we shall try to leave some computers for use until later, but by then we hope you will have a good idea of how to solve the problem below. During this time you will be able to find the following files in the usual places, or by following links from the course web page:

- All the files as were made available for the practical course.
- The file `EXAM.ZIP` which contains the C++ version of the Coco/R system, and its support files. If you unzip this file you will be able to run GenMake, QEdit, UltraEdit and Coco/R using your Favourite Language in the way that is, hopefully, familiar by now.
- Borland and Visual C++ compilers, which you can invoke in the usual way. You will not need to log on to a server to use these compilers; they are on the local C: drive.
- The entire set of source files for The Book, in a directory system starting at `I:CSC301\TRANS\SOURCES`, unpacked as described in Appendix A. The files are arranged by chapter and by language - for example, the C++ versions of the sources for chapter 18 are in `I:CSC301\TRANS\SOURCES\CHAP18\CPP`. If you have trouble locating these files you are free to ask for help (but the information is all in the file `I:CSC301\TRANS\SOURCES\README.SRC`).

Most of the machines in the labs will work in exactly the way they have done this semester. A few are set up to show you the configuration you can expect to find tomorrow. It is suggested that you work at one of those machines for a short time to make sure you know what to expect. To work on these machines you proceed as follows

- Open up a DOS window following the `START -> HAMILTON UG` route as usual
- Give the command

  ```
  CONNECT   TESTxx   demo
  ```

  where `xx` is the two digit number for the machine (typically in the range `01` through `25`).
- This will then allow you to log onto the `J:` drive, where you will find the file `EXAM.ZIP` waiting for you.
- Give the command

  ```
  UNZIP   EXAM.ZIP
  ```

  to unpack the "exam kit".

From here on things should be familiar. you could, for example, log onto the `D:` or `J:` drive, use `UltraEdit or QEdit`, run `GenMake` ... generally have hours of fun.

But note that the exam set up has *no* connection with the outside world - no web browser, ftp client, telnet client, shared directories - not even a printer!

Today you may use the files and either the "usual" or the "exam" 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 any files from the D: drive, so that others are not tempted to come and snoop around to steal ideas from you.

(b) You are encouraged 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.

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. You have plenty of time in which to prepare and test really good solutions - go for it and good luck! Remember that you may not bring any papers or diskettes into the room tomorrow.

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 Monday

Before the start of the formal examinations 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 receive ordinary answer books for the preparation of answers, and an examination paper.

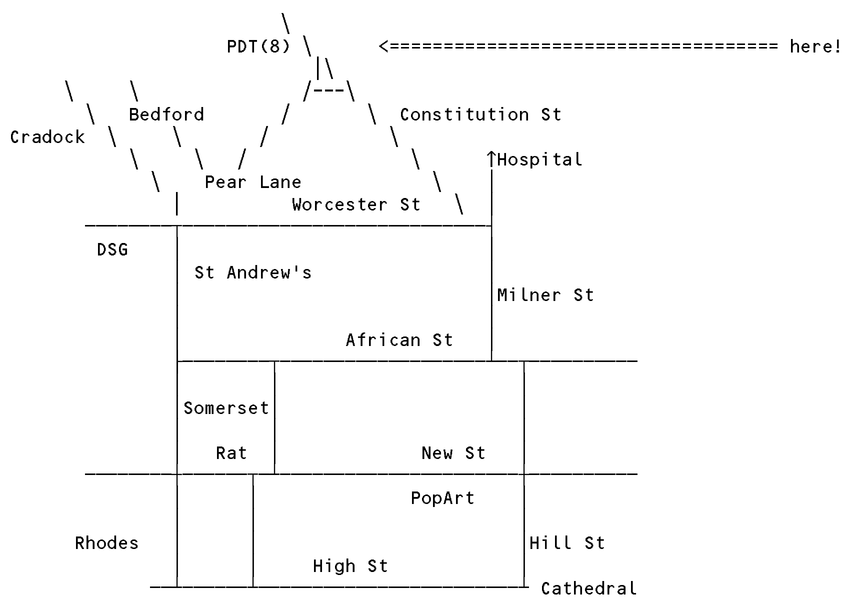
- You will be allocated to a computer and supplied with a `CONNECT` command for your own use. Once connected you will find an exam kit on the `J:` drive. This will contain the same Coco/R system and other files you have been given today, and in addition there will be "flat ASCII" machine readable parts of the examination paper itself, in files with names like `Q7.TXT` (Question 7). *There is no obligation to use a computer during the exam. You can answer on paper if you prefer - and yes, you can write in pencil if you prefer that.*

- A few copies of Chapter 12 of the text ("Using Coco/R - Overview") will be available as free information, should you need them.

- 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: drive back to the server. This will be explained tomorrow.

Cessation of Hostilities Party

As previously mentioned, Sally and I would like to invite you to an informal end-of-course party at our house on 12 June. There's a map below to help you find your way there. It would help if you could let me know whether you are coming so that we can borrow enough glasses, plates etc. E-mail to p.terry@ru.ac.za. Time: from 18h30 onwards. Dress: Casual



And for my last trick - taken exactly from tomorrow's examination paper!

Section B [90 marks]

Please note that there is no obligation to produce a machine readable solution for this section. Coco/R and other files are provided so that you can enhance, refine, or test your solution if you desire. If you choose to produce a machine readable solution, you should create a working directory, unpack EXAM.ZIP, modify any files that you like, and then copy all the files back to the blank diskette that will be provided.

During the translators course we have made frequent use of the Cocol/EBNF notation for expressing production rules, and in particular the use of the {} and [] meta-brackets introduced by Wirth in his 1977 paper. An example of a system of productions using this notation is as follows:

```
Home      = Family { Pets } [ Vehicle ] "house" .
Pets      = "dog" [ "cat" ] | "cat" .
Vehicle   = ( "scooter" | "bicycle" ) "fourbyfour" .
Family    = Parents { Children } .
Parents   = "Dad" "Mom" | "Mom" "Dad" .
Parents   = "Dad" | "Mom" .
Child     = "Margaret" | "Janet" | "Anne" | "Ntombizodwa" | "Ntombizanele" .
```

In analysing such productions and/or testing the grammar it has often been suggested that:

(a) they be rewritten without using the Wirth brackets, but using right recursive productions and an explicit ϵ, as for example

```
Home      = Family AllPets Transport "house" .
AllPets   = Pets AllPets | ε .
Transport = Vehicle | ε .
Pets      = "dog" PussyCat | "cat" .
PussyCat  = "cat" | ε .
Vehicle   = ( "scooter" | "bicycle" ) "fourbyfour" .
Family    = Parents Offspring .
Offspring = Offspring Children | ε .
Parents   = "Dad" "Mom" | "Mom" "Dad" .
Parents   = "Dad" | "Mom" .
Child     = "Margaret" | "Janet" | "Anne" | "Ntombizodwa" | "Ntombizanele" .
```

(b) a check be made to see that all the non-terminals have been defined (this does not occur in the above case - `Children` is undefined);

(c) a check be made to see that each non-terminal is defined only once (this does not occur in the above case - there are two rules for `Parents`);

(d) a check be made to see that each non-terminal (other than the goal) must appear on the right side of at least one production (this does not occur in the above case; `Child` defines a non-teminal which does not appear in any other production).

As a useful service to others who might take this course in future years (and hopeful that, although you might encourage others to do so, you are not forced to do so yourself!), develop a system using Coco/R that will carry out the above transformations and checks.

Here are some suggestions for deriving a complete system:

(a) In the exam kit (`EXAM.ZIP`) you will find the executables and support files for Coco/R, as used in the practical course. There is also the skeleton of a grammar file `EBNF.ATG` with suitable definitions of character sets and tokens similar to those you have seen before.

(b) It should be obvious that you will need to set up a "symbol table". In the exam kit is supplied the skeleton of an "include file" `EBNF.H`, which has a rudimentary form of table that you might like to extend. The template list handler file `LIST.H` familiar from earlier practicals has also been supplied.

(c) Inventing additional non-terminal names (without clashing with others that might already exist) might be done with the aim of deriving, for example

```
Home = Family HomeSeq1 HomeOpt2 "house" .
```

(d) In the exam kit will be found some other example data and production sets to assist in your development of the system, and an executable derived from a model solution.

(e) After converting a set of productions from EBNF to BNF, try converting the BNF productions once again to check that your system works consistently.

```
COMPILER EBNF $XCN
/* Convert a set of EBNF productions to use BNF conventions, and carry out
   some rudimentary checks on their being properly defined.
   YOUR IDENTITY HERE */

#include "ebnf.h"

CHARACTERS
  cr       = CHR(10) .
  lf       = CHR(13) .
  letter   = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz" .
  lowline  = "_".
  digit    = "0123456789" .
  noquote1 = ANY - "'" - cr - lf .
  noquote2 = ANY - '"' - cr - lf .

IGNORE CHR(9) .. CHR(13)

COMMENTS FROM "(*" TO "*)"  NESTED

TOKENS
  nonterminal = letter {letter | lowline | digit} .
  terminal    = "'" noquote1 {noquote1} "'" | '"' noquote2 {noquote2} '"' .

PRODUCTIONS
  EBNF
  =                                 (. TABLE_InitTable(); .)
    { Production }
    EOF .

  Production
  =                                 (. char Name[MaxName];
                                       int i; .)
    SYNC nonterminal                (. LexString(Name, sizeof(Name) - 1);
                                       TABLE_Add(Name, i);
                                    .)
    WEAK "="                        /* obviously more needed here */
    SYNC "."
    .

END EBNF.
```

```
// Various common items for grammar handler

#ifndef EBNF_H
#define EBNF_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <ctype.h>
#include <limits.h>

#define  boolean  int
#define  bool     int
#define  true     1
#define  false    0
#define  TRUE     1
#define  FALSE    0
#define  maxint   INT_MAX

#if __MSDOS__ || MSDOS || WIN32 || __WIN32__
#  define  pathsep '\\'
#else
#  define  pathsep '/'
#endif

static void appendextension (char *oldstr, char *ext, char *newstr)
// Changes filename in oldstr from PRIMARY.xxx to PRIMARY.ext in newstr
{ int i;
  char old[256];
  strcpy(old, oldstr);
  i = strlen(old);
  while ((i > 0) && (old[i-1] != '.') && (old[i-1] != pathsep)) i--;
  if ((i > 0) && (old[i-1] == '.')) old[i-1] = 0;
  if (ext[0] == '.') sprintf(newstr,"%s%s", old, ext);
    else sprintf(newstr, "%s.%s", old, ext);
}

const int MaxName = 50;

typedef struct {
  char name[MaxName];
  // you may need other members
} ENTRIES;

const int TABLE_MaxEntries = 1000;            // limit on table size
static int TABLE_NEntries;                    // number of active entries
static ENTRIES TABLE_Table[TABLE_MaxEntries]; // the table itself

static void TABLE_InitTable() {
// Initialise table
  TABLE_NEntries = 0;
}

static void TABLE_Add (char *name, int &position) {
// Attempt to add name to the table, and return position as the
// index where it was either added previously or added by this call
  if (TABLE_NEntries == TABLE_MaxEntries) {
    fprintf(stderr, "Table Overflow"); exit(1);
  }
  strcpy(TABLE_Table[TABLE_NEntries + 1].name, name); // sentinel
  position = 1;
  while (strcmp(name, TABLE_Table[position].name)) position++;
  if (position > TABLE_NEntries) TABLE_NEntries++;    // new entry
}

static void TABLE_ListProductions() {
// List all entries to standard output file
  for (int i = 1; i <= TABLE_NEntries; i++)
    printf("%s\n", TABLE_Table[i].name);
  printf("\n");
}

#endif /* EBNF_H */
```