EEL702: Major Test

November 27, 2013

Maximum Marks: 30

Note: The files for Questions 1 *and* 2 *are at* `http://web.iitd.ac.in/~sumeet/eel702`*. In case you cannot access the LAN, they will also be available on USB memory sticks.*

1. **A man of letters doesn't go into a shell: He scripts them!** Consider the problem of generating admission letters from a spreadsheet, and sending them over email. You are given a spreadsheet `admissions.xls` which looks like the following. The first column has the application number; the second has the name (with all parts of the name separated by a '.', without any space anywhere, for programming convenience); the third has the category: General, or Reserved; the fourth has the gender of the person, and the fifth, his/her email address. *Disclaimer: all the data below is purely ficticious, and resemblance to any person living, dead, or otherwise, is purely unintentional.* **[6]**

EE001	Hariharan.S	GEN	M	`hariiyer@gmail.com`
EE002	Kaustubh.S.Gadhvi	GEN	M	`kg.coep@gmail.com`
EE003	M.S.K.Padma	GEN	F	`padma_mariganti@yahoo.com`
EE004	Ranjana.Kumar	RES	F	`ranju1990@rediffmail.com`
EE005	Tshering.W.Tuithung	RES	M	`tuithung_guitar@gmail.com`

Write a shell script in `bash` or `tcsh` alone (no `perl`, `python`, or related software), using 'standard' UNIX tools alone, to generate admission letters, and mail them to each person. Keep in mind:

- Convert the spreadsheet `admissions.xls` into a text `admissions.csv` (Comma-Separated Values file).
- Assume the system to have the required shell `/bin/tcsh` or `/bin/bash`. Your shell script should run in the correct shell if the user simply writes the name of the script in the correct syntax. e.g.,
  `./admission_script admissions.csv`
  and not an explicit
  `bash ./admission_script admissions.csv`
- Do a proper syntax check: the script should take in only one parameter on the command line, else print the following error message: `usage:  admission_script <file>`
  `(Generate admission emails)`
- The gender of the person dictates the salutation as 'Mr.' or 'Ms.', and the corresponding hostel he/she will be put up in (read, 'which hostel he/she has to put up *with*'): women in Himadri, and men in Girnar. The category dictates the amount he/she will have to bring a crossed DD for: Rs.500/- for the reserved category (RES), and Rs.1000/- for the general category (GEN). For instance, the letter to Ms. Ranjana.Kumar `EE004.txt` will read:

```
Dear Ms.Ranjana.Kumar,
You will be accommodated in Himadri hostel
Please get a crossed DD for Rs.500/-
```

- Assume email software `elm` is present in the path. The syntax is:
  `elm -s "The subject line" abc@def`
  followed by the email text typed out on the terminal. You have to use it suitably in the shell script.

2. **Totally@C?** This apparently simple piece of C code `hack.c` just refuses to do what is intended. You have to take a character input using `scanf` alone, and yet, get the program to do its intended job. **[4]**

```
#include <stdio.h>
int main(void)
{
char c;
char string[30];
do{
printf("please enter any string: ");
scanf("%s",string);
printf("‘q’ to quit, other character to continue: ");
scanf("%c",&c);
}while(c != ’q’);
return 1;
}
```

3. **A hard software question... firm answers, please!** consider a DFA whose input alphabet Σ is the set of all English lower-case letters. Discuss how you will implement this using Flip Flops, and other digital components. No stories, please! **[3]**

4. (a) What are the goals of semantic analysis during compilation? Why can these not be achieved during parsing? **[2]**

   (b) Give the output of the following piece of code assuming (i) static scoping and (ii) dynamic scoping. You should explain your answers. **[3]**

```
int a = 12;
int b = 5;
void mult(){
        printf("%d\n",a*b);
}
void rec1(){
        int a = 1;
        mult();
}
void rec2(){
        int b = 7;
        rec1();
}
mult();
rec1();
rec2();
```

5. (a) State the key differences between the imperative and declarative programming paradigms. Give one advantage of each paradigm. **[2]**

   (b) State and describe two key features of object-oriented programming that distinguish it from procedural programming. **[2]**

6. (a) What are the three elements of formal verification? Why is temporal logic useful in this context? How might UML be useful for formal verification of a software system? **[2]**

   (b) What are safety, liveness, and fairness? Give one example of each kind of property, stated both in English and in temporal logic. **[2]**

   (c) Show (with derivation/justification) how the ‘always’ ($\Box$) and ‘sometime’ ($\Diamond$) operators can be rewritten in terms of only the ‘next’ ($\bigcirc$) and ‘until’ ($\mathcal{U}$) operators. **[2]**

7. Describe the functionality of the Map and Reduce procedures. Why are they useful for distributed computing? **[2]**