IS MSc Artificial Intelligence Programming II

Exercise 4 (Issued: week 4)

First try out a demo. Bring up a terminal window and type the following to the Unix prompt:

```
% pop11 +gblocks
```

Follow the instructions of the program.

Read Chapters 3,4,5, and 6 of the Pop-11 Primer. Do not worry about not understanding things I haven't yet covered in lectures.

For these exercises you will need a telephone list in the following form:

[[mary jones] 479625 [fred smith] 249371 [mike fret] 567923]−>telephones;

Note Your code should work for any telephone list in the right format, not just for the one above, or those of length 3!

You may find the procedures **hd**, **tl**, **length** useful.

1. Write four different versions of a procedure named print_phonelist that takes a telephone list as argument and prints out on a separate line each person's name and their telephone number. The procedure should not return a result. The four versions should make use of:

   (a) **repeat** ... **times**... **endrepeat**
   Hint: Use **hd** to get at the first element in a list, and **tl** to shorten the list. Each time round the loop we need to get at two elements (the name and the number), and to shorten the list by two, ready for the next time round. How many times must this be done?

   (b) **for** ... **in** ... **do** ... **endfor**
   Hint: You may find the procedures **isinteger** and/or **islist** useful, since they enable you to easily tell lists from integers.

   (c) **until** ... **do** ... **enduntil**
   Hint: This is similar to (a), but you need to think about how to tell when you have finished.

(d) **while** ... **do** ... **endwhile**
Hint: This is very similar to (c).

2. Write two different versions of a procedure get_name that given a telephone number and telephone list as arguments returns the corresponding name as its result e.g.

   get_name(249371,telephones)=>
   ** [fred smith]

   If the number is not in the telephone list then the procedure should return a list [no luck].

   One version should use the matcher, the other should use any looping construct.

3. Write a procedure named get_first_name that expects the same arguments as get_name (in Question 2) but simply returns the person's first name. The new procedure should call get_name as a sub-procedure.

4. Write a procedure named average that takes a single argument, a telephone list, and returns the arithmetic average (to the nearest whole number) of the telephone numbers it finds!

5. Write a procedure named find_average_person that takes a telephone list as argument and returns either the name of the first person who happens to have the same telephone number as the average, or if such a person cannot be found returns the list [no luck] as its result.

6. Write a procedure named get_nums of no arguments that makes use of readline to interactively build up a telephone list in the form above and when complete return it as its result. When the user types "no more" it should stop e.g.

   get_nums()->newlist;
   ** [name]
   ? mary haddock
   ** [number]
   ? 12345
   ** [name]
   ? jack sprat
   ** [number]
   ? 22222
   ** [name]
   ? no more
   newlist=>
   ** [[jack sprat] 22222 [mary haddock] 12345]