UNIVERSITY OF VICTORIA
EXAMINATIONS APRIL 1999

SENG 265 S01
SOFTWARE ENGINEERING I

NAME .... **REG. NO.** ....

Instructor: R. N. Horspool
Duration: 3 hours

TO BE ANSWERED ON THE EXAMINATION PAPER.

STUDENTS MUST COUNT THE NUMBER OF PAGES IN THIS EXAMINATION PAPER BEFORE BEGINNING TO WRITE, AND REPORT ANY DISCREPANCY IMMEDIATELY TO THE INVIGILATOR.

THIS QUESTION PAPER HAS NINE PAGES INCLUDING THIS COVER PAGE.

ATTEMPT ALL FIVE QUESTIONS. THE QUESTIONS HAVE EQUAL WEIGHT.

AIDS (BOOKS, NOTES, CALCULATORS) ARE PERMITTED.

Question	**Score**
1	
2	
3	
4	
5	
TOTAL	

Question 1

The function `longestPalindrome` shown in Figure 1 is intended to search a character string (its first argument) and find the longest palindrome embedded in that string; the position and length of that longest palindrome are returned via the second and third parameters. For example, the calling code

```
char *text = "a_radar_beam";  int start, length;
longestPalindrome(text, &start, &length);
printf("pos=%d len=%d\n", start, length);
```

should print the line "`pos=1 len=7`". It corresponds to the 7 character substring `"_radar_"`.

```
static int extendPalindrome( char *s, int p ) {
    int len = 1;
    while(p > 0 && s[p-1] == s[p+len]) {
        p--;
        len += 2;
    }
    return len;
}

void longestPalindrome( char *s, int *pos, int *len ) {
    int currPos, currLen, newLen;
    *pos = 0; *len = 0;
    for( currPos=0; currPos<strlen(s); currPos++ ) {
        newLen = extendPalindrome(s,currPos);
        if (newLen > *len) {
            *len = newLen;
            *pos = currPos;
        }
    }
}
```

Figure 1: An Imperfect longestPalindrome Function

(a) Fill in the following table with results produced by the function on the given inputs.
IMPORTANT : if the result of the function is *undefined* (due to an execution error) you must say so.

Input argument	**Results of function call**	
s	**pos**	**len**
`"abcdef"`		
`"aaa"`		
`"XabbaY"`		
`""`		

(b) You might have observed that the extend Palindrome function accesses its argument string at a position beyond its rightmost character. Is this an error? If not explain, and if it is an error, provide a correction.

Line	Your explanation or your correction

(c) You should have observed that the function does not return the correct positions. Propose *one* correction (an insertion of one statement, a deletion of one statement or a change to one statement) which causes the positions to be correct.

Line	Your correction

(d) You should also have observed that the function does not find all the palindromes that it should. There does not appear to be a simple one-line correction to this fault. Explain in a few sentences what change(s) you would make to the code so that it did locate all the desired palindromes.

Line(s)	Your description of what needs to be changed

Question 2

In the table below, write the exact output of the Perl program shown in Figure 2. **Important Note**: space characters in the string (the value assigned to `$string`) are shown as underscores for legibility.

Note: After a successful pattern match, the `$&` special variable contains the text which matched the pattern.

Line	Output
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

```
#!/public/bin/perl
@patList = ('ai...',       '\+.*',        '[ai][^ai]',
           'al*',          '\w+',         '...$',
           '^[^a-z_]*',    '\s\w\w',      '\w{2,3}\.',  '\W..\W' );
$string = "The_rain_in_Spain_falls_mainly_in_the_plain._++*!!";
foreach $pat (@patList) {
   if ($string =~ /$pat/) {
      print "--$&--\n";      ## printing line i
   } else {
      print "no match\n";    ## printing line i
   }
}
```

Figure 2: Perl Program for Question 2

Question 3

Consider the `Makefile` shown in Figure 3. Assume that the current directory contains the following files with the file permissions and timestamps shown.

```
Makefile      -rw-r--r--    Jan 1  18:40
alan.x        -rw-r--r--    Mar 2  10:29
beth.x        -rw-r--r--    Mar 2  10:32
chuck.x       -rw-r--r--    Mar 7  09:11
chuck.o       -rw-r--r--    Mar 7  11:22
dana.i        -rw-r--r--    Mar 4  12:00
edward.i      -rw-r--r--    Mar 4  13:17
```

Fill in the exact output produced by `make` in the table below. Note that the commands are executed one after the other; files created in a preceding step are assumed to still exist (unless explicitly deleted by one of the actions of the Makefile).

Helpful Notes:

- When `make` executes a shell command (e.g., "`gcc -g -c francis.c`"), it prints that command.
- If `make` is asked to create target `X` that already exists and is up-to-date, then `make` prints the message "`X is up to date`".
- If `make` is asked to create target `X` and the `Makefile` contains no instructions for creating `X`, then `make` prints the message "`Don't know how to make target X`".
- The command "`touch somefile`" causes the timestamp on the named file (`somefile`) to be changed to the current date and time.
- The command "`tar cvf aa b c d`" creates a tar archive file named `aa`, where the members of the archive are the files `b`, `c` and `d`.

```
## Makefile for Question 3
.SUFFIXES: .x .tar .tgz
.x.o:
        cp $*.x $*.c
        gcc -g -c $*.c
        rm $*.c
.tar.tgz:
        gzip < $*.tar > $*.tgz
        rm $*.tar
SRCS = alan.x beth.x chuck.x \
       dana.i edward.i
OBJS = alan.o beth.o chuck.o
prog: $(OBJS)
        gcc -g $(OBJS) -o $@
alan.o: dana.i
beth.o: dana.i edward.i
charles.o: edward.i
clean:
        rm -f *.o
arch.tar: $(SRCS)
        tar cvf arch.tar *.x *.i
arch.tgz: arch.tar clean
```

Figure 3: Makefile for Question 3

	Command(s)	Output printed by make
1	`make`	

2	`touch dana.i`
	`make`	

3	`make arch.tgz`	

4	`make clean`
	`make beth.o`	

5	`touch beth.c`
	`make arch.tgz`	

Question 4

Consider the Perl program shown in Figure 4. It For each line of input that it reads, it can generate some lines of output. The program is to be run, reading the 5 lines of input shown in the first column of the table below. Write the output lines printed by the program alongside the corresponding input line in the table below..

Input Line	The Program Output
`A = 99 + B;`	
`B = 2 * A - B;`	
`C = A + (A/2);`	
`B = C - 1;`	
`print(B, D);`	

```
#!/public/bin/perl
$i = 0;
while( <STDIN> ) {
    $line = $_;
    $i = $i + 1;
    if ($line =~ s/^\s*([A-Za-z]\w*)\s*=//) {
        $target = $1;
    } else {
        $target = undef;
    }
    while($line =~ s/[A-Za-z]\w*//) {
        if (!defined($d{$&}) && $& ne 'print') {
            print "$i: possibly uninitialized variable $&\n";
            $d{$&} = 0;
        }
        $u{$&} = $i;
    }
    if (defined($target)) {
        if (defined($d{$target}) && !defined($u{$target})) {
            print "$d{$target}: useless assmt to $target\n";
        }
        $d{$1} = $i;
        $u{$1} = undef;
    }
}
```

Figure 4: Perl Program for Question 4

Helpful Notes:

- After a successful pattern match, the special variable `$1` contains the text that matched the 1st parenthesized expression in the pattern. (Similarly for `$2` and so on.)
- After a successful pattern match, the special variable `$&` contains the text that matched the entire pattern.
- The function call `defined($v)` will return true if `$v` has the value `undef` and will return false otherwise.
- If a hash table `h` does not contain an entry with `"k"` as a key, then looking up `$h{"k"}` will create a new entry with `"k"` as the key and `undef` as the associated value.

Question 5

Suppose that executing the Unix command "`ls -l`" in the current directory produces a *partial* listing of files as shown below. Further suppose that your userid is `fmulder`. The current directory is *not* your home directory.

```
-rw-r--r-- 1 fmulder       233   Nov 4 12:23  Makefile
drwxrwxrwx 2 fmulder      1536   Nov 6 07:51  RCS
-rw-r--r-- 1 fmulder      4324   Nov 8 09:54  input.h
-rw-r--r-- 1 fmulder   3483456   Nov 8 09:35  prog.c
-rw-r--r-- 1 fmulder      3864   Nov 8 09:57  output.h
... and many more files
```

For each of the ten desired actions given in the table below, write appropriate Unix command(s) that accomplishes the action.

Desired Operation or Task	Unix Command(s)
Copy the directory `RCS` (and all its contents) into your home directory.	
Check out, using RCS, read-only copies of the current versions of all files that have the suffix '`.c`' (once checked out).	
Start compiling the file `prog.c` as a background job.	
Terminate (kill) the background job started in the preceding step.	
Rerun a compilation of the file `prog.c` as a background job, but this time saving all error messages in a new file named `ERRS`.	

Desired Operation or Task	Unix Command(s)
Change your `$path` variable to include the additional directory `/public/bin`. (If your Unix commands assume a different command shell than `csh`, please state which shell it is.)	
Search files `prog.c` and all '`.h`' files in the current directory for all lines that contain the word '`typedef`'.	
Change the file access permissions on all files in the current directory so that they cannot be read by any other user.	
Change the access permissions on the RCS subdirectory so that other users cannot visit or retrieve files from that directory.	
Mail a uuencoded and compressed version of the file named `a.out` to a user with e-mail address `dscully@xfiles.com`.	

Have a great summer everyone!

<u>END</u>