UNIVERSITY OF VICTORIA
EXAMINATIONS DECEMBER 1998

SENG 265 F01
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 SIX 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 `split` shown in Figure 1 is intended to split a long line (its first argument) so that its length does not exceed `maxlen` (its second argument) and can therefore be printed in a text window with width `maxlen`. The function is supposed to split the line at the last suitable space character so that a word is not split in the middle. The function changes the line passed as its argument by truncating that string and it returns the part of the string that has been chopped off as its result. For example, the calling code

```
char text[100] = { "a piece of nonsense text" };  char *r;
r = split(text, 15);
printf("%s\n", text);
```

should print the line "`a piece of`" and leave `r` pointing at the string "`nonsense text`".

(a) Fill in the following table with results produced by the function on the given inputs:. **IMPORTANT NOTES**: to improve legibility, space characters are shown as dots in the table; if the result of the function is *undefined* you must say so.

Input arguments		Results of function call	
line	**maxlen**	**line**	**result**
"1.2.3.4.5.6.7"	10		
"............"	10		
""	10		
"abcdefghijklm"	10		

(b) Propose *one* correction (an insertion of one statement, a deletion of one statement or a change to one statement) which eliminates undefined behaviour from the function.

Line	Your correction

```
1     char *split( char *line, int maxlen ) {
2        int len = strlen(line);
3        int i, pos;
4        if (len <= maxlen) return NULL;
5        for( i=0;  i<maxlen;  i++ ) {
6           if (line[i] == ' ') pos = i;
7        }
8        line[pos] = '\0';
9        return &line[pos+1];
10    } /* There is a bug in this code !! */
```

Figure 1: An Imperfect split Function

Question 2

In the table below, write the exact output of the Perl program shown in Figure 2.

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 = ('bc',          'bbc*',       '(cc)|(bb)',
            'bc*',         '\w+',        '\w+$',
            '([bc]|[^*])+', '\w\wb.\d+', '\w{2,3}\+',     '\W..' );
$string = "abbccc*123+";
$i = 1;
foreach $pat (@patList) {
   if ($string =~ /$pat/) {
      print "--$&--\n";       ## printing line i
   } else {
      print "no match\n";     ## printing line i
   }
   $i++;
}
```

Figure 2: Perl Program for Question 2

Question 3

Consider the `Makefile` shown in Figure 3. Assume that the current directory contains no files other than `Makefile`. Assume that the five `make` commands below are executed in the order given and that any files created by `make` in a preceding step still exist (with the same time stamps) when the next command is executed. Fill in the exact output produced by `make` in the table below.

	Command	Output printed by make
1	`make`	
2	`make g`	
3	`make clean`	
4	`make f`	
5	`make f`	

Notes:

- The Unix command "`date`" prints the current date and time on its standard output stream.
- The Unix command "`sleep 1`" simply causes a delay of 1 second and does nothing else.
- When `make` executes a shell command (e.g., "`date > f`") 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`".

```
## Makefile for Question 3
.SUFFIXES: .a .b
.a.b:
        cp $*.a $*.b
g: g0.b g1.b
        cat g0.b g1.b > g
g0.b:
        date > g0.b
g1.b:
        date > g1.b
clean:
        rm -f *.a *.b f g
f: f0.b f1.b
        cat f0.b f1.b > f
        sleep 1
        date > f1.a
f0.b:
        date > f0.b
f1.b: g0.b
        date > f1.b
```

Figure 3: Makefile for Question 3

Question 4

Consider the Perl program shown in Figure 4. For each line of input that it reads, it can generate some lines of output. Write these output lines 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);`	

Note: 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.)

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

Figure 4: Perl Program for Question 4

Question 5

For each operation described in the first column of table below, write the names of one or more Unix commands that should be appropriate. Use only commands taken from the list given in Figure 5. If more than one command might be appropriate, you must list *all of them*.

Desired Operation or Task	Possibly Appropriate Commands
Rename a directory	
Execute a text file containing Perl source code	
Create a new text file	
Convert upper-case letters to lower-case in a text file	
Find all files in current directory which contain the text "`subr1`"	
Check the setting of the `DISPLAY` environment variable	
Discover why a C program runs slowly	
Combine several files into a compressed archive file	
Combine several text files into a single file	
Send a file to a different computer	

`bg`	`ftp`	`pr`
`cat`	`gcc`	`printenv`
`cc`	`gdb`	`prof`
`cd`	`gprof`	`rm`
`chmod`	`grep`	`sccs`
`compress`	`gzip`	`set`
`cp`	`kill`	`setenv`
`date`	`mail`	`tar`
`dbx`	`make`	`tcov`
`fg`	`mv`	`tr`
`fmt`	`perl`	`uuencode`

Figure 5: Unix Commands to be used in Question 5

END