Duration: **50 minutes**
Aids Allowed: **None**

Student Number: |_|_|_|_|_|_|_|_|_|_|_|

Last Name: SOLUTION

First Name: _____

TA: _____ **Instructor:** Reid

---

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below *carefully*.)

This midterm test consists of 5 questions on 5 pages (including this one), plus the aid sheet. *When you receive the signal to start, please make sure that your copy of the test is complete.* Extra space was left for each of the programming questions. Please indicate clearly the part of your work that should be marked.

IMPORTANT: You do not need to include the "#!" line in Bourne shell or Python programs you are asked to write. In C programs, you do not need to add "#include" lines, or do error checking unless the question requires it, or the program would not function correctly given valid input without error checking.

MARKING GUIDE

1: _____/ 5
2: _____/ 3
3: _____/ 8
4: _____/ 7
5: _____/ 6

TOTAL: _____/29

Good Luck!

Question 1. [5 MARKS]

Part (a) [1 MARK]

Write the Unix command that needs to be executed to make the shell script `myprog` executable. Assume `myprog` is in the current working directory.

`chmod u+x myprog` or `chmod 700 myprog` or some variation.

Part (b) [2 MARKS]

Briefly explain what happens when the following line is executed in the Bourne shell.

```
mkdir ~/SBS > /dev/null 2>&1
```

A directory called SBS is created in the user's home directory. If there is any output, it is redirected to /dev/null, and won't appear on the screen or in a file.

Part (c) [1 MARK]

Write one line of Bourne shell to print out an asterisk.

echo '*'

Part (d) [1 MARK]

Write one line of Bourne shell to print out the name of the variable given below and its value.

```
foo=10
```

```
echo ``$foo'' $foo
```

Question 2. [3 MARKS]

Fill in the Perl regular expressions so that when the expression is applied to each of the three input strings listed at the top, the corresponding value is assigned to `$1`

			input	"10"	"-15"	"15.4"
/(	`\d+`	)/	`$1 ==`	"10"	"15"	"15"
/(	`-?\d+`	)/	`$1 ==`	"10"	"-15"	"15"
/(	`-?[\d\.]+`	)/	`$1 ==`	"10"	"-15"	"15.4"

Question 3. [8 MARKS]

When the `sbs` program from assignment 1 is run as `sbs init`, it creates a directory called `SBS` in the user's home directory.

Part (a) [2 MARKS]

Briefly describe the two main test cases required to test `sbs init` and state the expected outcome of each test.

- If the directory `~/SBS` does not exists, `sbs init` creates it.
- If the directory `~/SBS` exists, `sbs init` prints an error message or does nothing.

Part (b) [6 MARKS]

Write a Bourne shell program to run `sbs init` and test the above two test cases. You may assume that the `sbs` program can be run from any directory. Your program may not make any assumptions about the existence of any files or directories other than the `sbs` program and the standard Unix programs or commands must exist and be executable. The program should print appropriate messages so that the tester understands the outcome.

```
#!/bin/sh

# Remove existing repository if there is one
rm -r ~/SBS

echo "init 1: Testing sbs init when no repository exists"
sbs init

if test -d ~/SBS
then
     echo "   ~/SBS created"
else
     echo "ERROR ~/SBS not found"
fi

echo "init 2: Testing sbs init when repository already exists"

# Make sure the directory exists
mkdir -p ~/SBS

echo "Expecting error message"
sbs init
```

Question 4. [7 MARKS]

Write a C program that will print out the names of all of the files executable by the user in the directory given by the first argument to the C program. Note that the st_mode field of the stat struct holds the permissions of the file. If the variable `m` holds the permissions for a file, the following expression evaluates to true if the user execute permission is set for `m`: `S_IXUSR & m`.

```
int main(int argc, char *argv[])
{
    DIR *dir = opendir(argv[1]);
    struct dirent *entry;
    struct stat sbuf;

    while(entry = readdir(dir)) {
        if(stat(entry->d_name, &sbuf) == 0) {
            if(S_ISREG(sbuf.st_mode)) {
                if(S_IXUSR & sbuf.st_mode) {
                    printf("%s\n", entry->d_name);
                }
            }
        }
    }
}
```

Question 5. [6 MARKS]

Complete the following Perl program. The program reads one line from standard input. The program prints out all the numbers on the line except the largest and the smallest. The order of the printed numbers does not matter. One mark will be deducted from a correct solution that has more than one loop. A bonus mark will be awarded for a correct solution with no loops.

For example, if the input is "5.6 5.4 5.7 5.7 5.6" the output is "5.6 5.7 5.6" (in any order).

```
# Solution 1
$line = <STDIN>;
@scores = split(/ /, $line);
my $max_index = 0;
my $min_index = 0;

# find the max and min
for(my $i = 0; $i < @scores; $i++) {
  if($scores[$i] > $scores[$max_index]) {
    $max_index = $i;
  }
  if($scores[$i] < $scores[$min_index]) {
    $min_index = $i;
  }
}
for($i = 0; $i < @scores; $i++) {
  if($i != $min_index && $i != $max_index) {
    print "$scores[$i] ";
  }
}
print "\n";

# Solution 2
$line = <STDIN>;
@scores = split(/ /, $line);
@sorted = sort {$a <=> $b} @scores;
$str = join(' ', @sorted[1..$#sorted-1]);

print "$str\n";
```