Duration: **50 minutes**
Aids Allowed: **One 8.5 x 11 inch paper**

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

Last (Family) Name: SOLUTION

First (Given) Name(s): _____

Tutorial Section: (circle one)

BA–1180	BA–B026
Xiaoyang Guan	Josh Bronson
	Andres Lagar Cavilla

---

*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 term test consists of 5 questions on 6 pages (including this one), printed on one side of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete.*

Answer each question directly on the test paper, in the space provided. If you need more space for one of your solutions, use the extra page at then end. *Indicate clearly the part of your work that should be marked.*

General Hint: We were careful to leave ample space on the test paper to answer each question.

MARKING GUIDE

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

TOTAL: _____/29

Good Luck!

Question 1. [5 MARKS]

The current working directory contains 3 files: `beer`, `coffee`, and `tea`. The contents of each file are shown below:

`beer`

```
barley
```

`coffee`

```
beans
```

`tea`

```
leaves
```

The program listing for `drinks.sh` is shown below (’ is a single quote, and ‘ is a back-quote):

```
#!/bin/sh

echo "Part 1:"
for x in $*
do
        echo $x
done

echo "Part 2:"
for x in *
do
        echo $x
done

echo "Part 3:"
echo "cat $1"
echo ’cat $1’
echo ‘cat $1‘
```

Fill in the missing parts of the output when the program is run as
`drinks.sh tea juice wine`

```
Part 1:
juice
wine

Part 2:
beer
coffee
tea

Part 3:
cat tea
cat $1
leaves
```

Question 2. [5 MARKS]

Program A

```
int main() {
    int result;
    printf("Q\n");

    if((result = fork()) > 0) {
        printf("R\n");
    } else if(result == 0) {
        printf("S\n");
        exit(0);
    } else {
        perror("fork");
    }
    printf("T\n");
    return 0;
}
```

Program B

```
int main() {
    int result;
    printf("Q\n");

    if((result = fork()) > 0) {
        printf("R\n");
    } else if(result == 0) {
        printf("S\n");
    } else {
        perror("fork");
    }
    wait(0);
    printf("T\n");
    return 0;
}
```

For each of the following statements, circle the program or programs that the statement is applicable to. Note that the statements are worded to avoid making assumptions about the number of times each line of output is printed.

[A] [B] One or more Q's are always printed before an R.

A B All of the Ts are always printed last.

A B All R's are always printed before all S's.

[A] [B] It is possible for the child to become a zombie.

[A] B It is possible for the child to become an orphan.

Question 3. [7 MARKS]

Based only on the information in the statements below answer the following questions.

```
int main(){
    char a[10] = "pocket";
    char *p = malloc(10 * sizeof(char));
    char *r = "wocket";
    char *s;
```

Part (a) [1 MARK] How many bytes of memory are allocated on the stack after executing this code?

22

Part (b) [1 MARK] How many bytes of memory are allocated in the heap?

10

Part (c) [1 MARK] How many bytes of memory are allocated (or reserved) in global memory?

0 or 7 (for "wocket")

Part (d) [4 MARKS]

If the following code is executed after the code at the top of the page, what is the output? If an error occurs, state what the error is, assume that the offending line has no effect, and that execution continues. Read the code carefully!

```
    s = strncpy(p, "in my", 10);
    s[0] = 'o';
    s = &a[3];
    printf("%s\n", &a[3]);
    printf("%c\n", r[4]);
    printf("%s\n", p);
    r[0] = 's';
    printf("%s\n", r);
```

```
ket
e
on my
is is an error to assign to r because it points to a string literal.
```

Question 4. [5 MARKS]

Write a C program that reads from stdin and writes to stdout. The input is a single line, in the format of a `#include` line from a C program and the output is the name of the included file. Assume that the `#include` line has the correct syntax. You program only needs to handle file names enclosed in `<>`.

For example, if the input line was `#include <stdio.h>` the output would be `stdio.h`.

This is fairly straightforward if you use `strchr` *or* `index`*. It is a little more involved if you iterate over the string directly.*

```
int main()
{
    char line[80];
    char *ptr, *end;
    fgets(line, 80, stdin);

    ptr = strchr(line, '<');
    ptr++;
    end = strchr(ptr, '>');
    *end = '\0';
    printf("%s\n", ptr);

    return 0;
}
```

Question 5. [7 MARKS]

Part (a) [2 MARKS]

Write a shell command that writes all `#include` lines from all the files in the current working directory to a file called `output`. (This subquestion is not related to question 4.)

grep -h include *.c ¿ output

Part (b) [5 MARKS]

Complete the shell script below so that it does the following:

1. Reads lines from the `output` file produced in Part (a) above.
2. Extracts each included file name (using the program from question 4, which you can assume is called `getinc`.
3. Prints a message stating which of the directories in `incdirs` contains that file. (There may be more than one.)

You should assume that the lines in `output` are in the format that can be handled by `getinc`.

```
#!/bin/sh

incdirs="/usr/include /usr/local/include /sw/include ."
while read line
do
    str=‘echo $line | getIfile‘
    for d in $incdirs
    do
if [ -f $d/$str ]
then
    echo $str is in $d
fi
    done

done < output
```