Duration: **50 minutes**
Aids Allowed: **1 - 8.5x11 sheet**

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 7 pages (including this one). *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 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:	______/ 7
# 2:	______/ 6
# 3:	______/ 6
# 4:	______/ 9
# 5:	______/ 7
TOTAL:	______/35

Good Luck!

Question 1. [7 MARKS]

Part (a) [1 MARK]

Suppose the current working directory contains a shell script called `runit`. Explain why the following error message is produced when you try to run the program.

```
$ runit
$ runit: Command not found.
```

The shell doesn't know where to find the program runit, because the directory containing runit was not found in the PATH variable. Specifically, the current working directory is not in the PATH.

Part (b) [1 MARK]

Explain precisely how to fix the problem described in part (a).

Run the program as `./runit` *or add the current working directory to the PATH variable by running*

```
setenv PATH ${PATH}:.
```

Part (c) [2 MARKS]

Suppose the current working directory contains a shell script called `alsorun`. Describe the two possible explanations for why the following error message is produced when you try to run the program.

```
$ alsorun
$ alsorun: Permission denied
```

Either you do not have read permissions or execute permissions on the file.

Part (d) [2 MARKS]

Write 2 lines of C code that would result in a segmentation fault (i.e., attempting to access an invalid address).

```
int *i = NULL;
*i = 10;
```

Part (e) [1 MARK]

Rewrite your example in part (d), fixing the error. You must use exactly the same variable(s).

```
int *i = malloc(sizeof(int));
*i = 10;
```

Question 2. [6 marks]

Write a shell program that reads from standard input, concatenates the lines it reads into one long line, and prints the line to standard output. It also returns the number of lines that it read.

```
#!/bin/sh

args=""
count=0

while read arg
do
    args="$args $arg"
    count=`expr $count + 1`
done

echo $args
exit $count
```

Question 3. [6 MARKS]

Consider the following program. In your answers below, assume all processes terminate normally.

```
int main()
{
    int id;
    int i;
    printf("A %d\n", i);
    id = fork();
    if(id == 0) {
        i = 1;
        printf("B %d\n", i);
        exit(0);
    }

    id = fork();
    if(id == 0) {
        i = 2;
        printf("C %d\n", i);
    }
    printf("D %d\n", i);
    exit(0);
}
```

Part (a) [1 MARK]

How many processes, including the original one are created? 3

Part (b) [5 MARKS]

Give the output of this program in a valid order.

```
A 0
D 0
B 1
C 2
D 2
```

Question 4. [9 MARKS]

Part (a) [5 MARKS]

Consider the following C declarations. Write the type of the expressions that follow or `invalid` if it is not legal C. If the type is not a pointer, then give its value.

```
struct point {
    char *name;
    int y;
};
struct point p[3] = {{"Yuan", 1}, {"Ou", 2}, {"Andres", 3}};
struct point *q = &p[1];
```

	Type	**Value ()**
`(q+1)->name`	pointer to char	Andres
`q[0]->name`	invalid	
`p[2].y`	int	3
`p[1]`	struct point	”Ou”, 2”
`(p+1)->y`	int	2

Part (b) [4 MARKS]

Write a C program that takes a file name as an argument and prints to standard output the number of lines in the file and the number of characters in the file not including the newline characters.

```
int main(int argc, char **argv)
{
    FILE *fp = fopen(argv[1], "r");
    char line[256];
    int linecount = 0;
    int charcount = 0;

    while(fgets(line, 256, fp) != NULL) {
        linecount++;
        charcount += strlen(line);
    }
    printf("%d %d\n", linecount, charcount);
    return(0);
}
```

Question 5. [7 MARKS]

Complete the C function below. The argument is a string in the format of a `Makefile` target line. The function returns a string with the same target and prerequisites in the format of a target line from assignment 1.

The format of a `Makefile` target line is shown below. For the purposes of this question there is exactly one space between each of the elements of the line. There can be any number of prerequisites.

```
target : prereq1 prereq2 prereq3
```

The format of a target line from assignment 1 follows. Assume that only one space separates the elements of the line. There can be any number of prerequisites.

```
@ target prereq1 prereq2 prereq3
```

```
char *
transform(char *src)
{




    char *sptr, *eptr;
    int len = strlen(src) + 1;
    char *dest = malloc(len);

    if(src[0] != '@') {
        fprintf(stderr, "Invalid format: %s\n", src);
        return NULL;
    }
    sptr = &src[1];

    while(*sptr == ' ')
        sptr++;

    if((eptr = strchr(sptr, ' ')) == NULL) {
        strncpy(dest, sptr, strlen(src) + 1);
        strncat(dest, " : ", len - strlen(dest));
        return dest;
    } else {
        *eptr = '\0';
        eptr++;
        strncpy(dest, sptr, strlen(src) + 1);
        strncat(dest, " : ", len - strlen(dest));
        strncat(dest, eptr, len - strlen(dest));
        return dest;
    }
}
```

Note: There is a shorter answer if it is done character by character.

C functions for strings:

```
char *index(const char *s, int c);
char *strncat(char *dest, const char *src, size_t n);
char *strchr(const char *s, int c);
size_t strlen(const char *s);
int strncmp(const char *s1, const char *s2, size_t n);
char *strncpy(char *dest, const char *src, size_t n);
char *strstr(const char *haystack, const char *needle);
```

C functions for files and directories:

```
int closedir(DIR *dir);
int fclose(FILE *stream);
char *fgets(char *s, int n, FILE *stream);
FILE *fopen(const char *file, const char *mode);
int fprintf(FILE *stream, const char *format, ...);
char *getcwd(char *buf, size_t size);
DIR *opendir(const char *name);
struct dirent *readdir(DIR *dir);
int stat(const char *file_name, struct stat *buf);
void perror(const char *s);
```

C functions for processes:

```
pid_t fork(void);
pid_t wait(int *status)
pid_t waitpid(pid_t pid, int *status, int options);
```

```
struct stat {
  dev_t         st_dev;      /* device */
  ino_t         st_ino;      /* inode */
  mode_t        st_mode;     /* protection */
  nlink_t       st_nlink;    /* number of hard links */
  uid_t         st_uid;      /* user ID of owner */
  gid_t         st_gid;      /* group ID of owner */
  dev_t         st_rdev;     /* device type (if inode device) */
  off_t         st_size;     /* total size, in bytes */
  unsigned long st_blksize;  /* blocksize for filesystem I/O */
  unsigned long st_blocks;   /* number of blocks allocated */
  time_t        st_atime;    /* time of last access */
  time_t        st_mtime;    /* time of last modification */
  time_t        st_ctime;    /* time of last change */
};
```

The following POSIX macro functions are defined to check the file type (m is the st_mode field of the stat struct):

```
S_ISLNK(m) is symbolic link?
S_ISREG(m) regular file?
S_ISDIR(m) directory?
```

Shell comparison operators

`-d filename`	Exists as a directory
`-f filename`	Exists as a regular file.
`-r filename`	Exists as a readable file
`-w filename`	Exists as a writable file.
`-x filename`	Exists as an executable file.
`-z string`	True if empty string
`str1 = str2`	True if str1 equals str2
`str1 != str2`	True if str1 not equal to str2
`int1 -eq int2`	True if int1 equals int2
`-ne, -gt, -lt, -le`	For numbers
`!=, >, >=, <, <=`	For strings
`-a, -o`	And, or.