




# CSC209 Finals Review

- 
- Wednesday, August 22, 2018
  - 2:00 – 5:00 pm
  - Room EX300
  - NO study aids allowed
  - **OFFICE HOURS: Monday, Aug 20, 2018**
    - **BA3289, 4:00 – 7:00PM**

UNIVERSITY OF TORONTO  
Faculty of Arts and Science  
St. George Campus  
SUMMER 2018 EXAMINATIONS  
CSC209H1S  
3 hours  
No Examination Aids

Student Number: \_\_\_\_\_

Last Name: \_\_\_\_\_

First Name: \_\_\_\_\_

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 final examination consists of 24 questions on 22 pages. A mark of at least 35 out of 88 on this exam is required to pass this course, otherwise your final course grade will be no higher than 47%. When you receive the signal to start, please make sure that your copy of the examination is complete.

You are not required to add any **#include** lines, and unless otherwise specified, you may assume a reasonable maximum for character arrays or other structures. Error checking is not necessary unless it is required for correctness or specifically requested.

Good Luck!



Marking Guide	
Part I	/4
5	/3
6	/4
7	/2
8	/2
9	/4
10	/2
11	/4
12	/4
13	/2
14	/2
15	/4
16	/4
17	/4
18	/2
19	/9
20	/6
21	/4
22	/4
23	/8
24	/10
TOTAL	/88





## What to study?

- PCRS all modules
- Lectures and Labs
- Worksheets
- Textbooks



Let's Get Started

Consider the following C code segment that redirects standard output to the file my.file:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
int fd;
mode_t fd_mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;

if ((fd = open("my.file", O_WRONLY|O_CREAT, fd_mode)) == -1)
    perror ("Could not open my.file");
else {
    if (dup2(fd, STDOUT_FILENO) == -1)
        perror ("Could not redirect std output");
    close(fd);
}
```

For each of the following three cases, you are to draw a simple representation of the file descriptor table, indicating the meanings of the values in each indexed location.

After the open statement File Descriptor table	
0	stdin
1	stdout
2	stderr
3	my.file

After the dup2 statement File Descriptor table	
0	stdin
1	my.file
2	stderr
3	my.file


After the close statement File Descriptor table	
0	stdin
1	my.file
2	stderr



Consider the following C program:

```
#include <stdio.h>

int main (void) {
    pid1 = fork();
    pid2 = fork();
    printf ("pid1 = %d, pid2 = %d\n", pid1, pid2);
    return 0;
}
```



Draw the resulting process tree relationship (and assign each process a name like A, B, ... )

The original process, A is the parent of a child process B. That child begins executing before A and creates its own child C. After B and C, the original parent process A creates another child process D. The process tree structure looks like:

A ----> B ----> C  
|  
D

```
#include <stdio.h>
int main (void) {
    pid1 = fork();
    pid2 = fork();
    printf ("pid1 = %d, pid2 = %d\n", pid1, pid2);
    return 0;
}
```

Using the process names shown in part (a), show what the program prints, where you label each output line with the name of the process that generated it. For the purpose of this question, a.) assume a simple, sequential process id numbering scheme, and b) assume that the most-recently created process is the process that runs on the processor until it terminates, and then the next-most-recently created process run, ... .

A runs to fork B. A's pid1 is the pid of B. B's pid1 is 0. B runs to fork C. B's pid2 is the pid of C. C's pid2 is 0 and its pid1 is 0, as pid1 is copied from B. C runs to completion and prints first. B runs to completion and prints. A runs to fork D. A's pid2 is the pid of D. D's pid2 is 0. D's pid1 is the pid of B, as pid1 is copied from A. D runs to completion and prints. A runs to completion and prints.

First, C print 0 0

Then B prints 0 20334

Then D prints 20333 0

Then A prints 20333 20335

```
#include <stdio.h>
int main (void) {
    pid1 = fork();
    pid2 = fork();
    printf ("pid1 = %d, pid2 = %d\n", pid1, pid2);
    return 0;
}
```

With the given variable declarations and initializations, give the numeric value of the following expressions. "bad value" and "undefined value" are also possible answers.

```
int i = 3, j = 5, *p = &i, *q = &j, *r;
```

a) `p == &i`

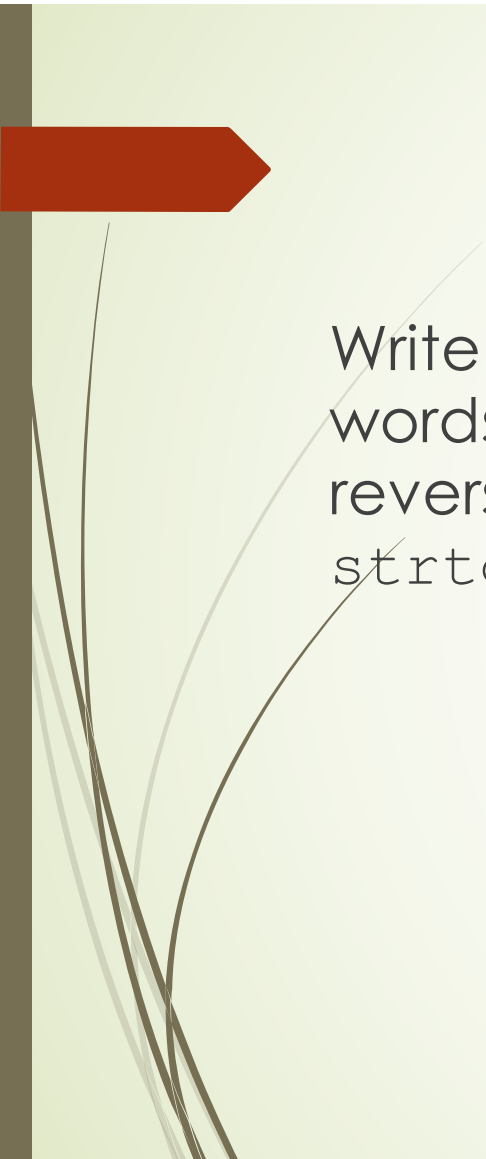
**1**

b) `**&p`


**3**

c) `*(r = &j) *= *p`

**15**



Write a program that prompts the user to enter a series of words separated by single spaces, then prints the words in reverse order. Read the input as a string, and then use `strtok` to break it into words.



```
#include <stdio.h>
#include <string.h>
#define WORDS_LEN 1024
void print_reverse(char *words);
int main(void) {
    char words[WORDS_LEN+1];
    printf("Enter a series of words: ");
    fgets(words, sizeof(words), stdin);
    printf("In reverse order:");
    print_reverse(words);
    printf("\n");
    return 0;
}
void print_reverse(char *words) {
    char *s = strtok(words, " \t\n");
    if (s != NULL) {
        print_reverse(s + strlen(s) + 1);
        printf(" %s", s);
    }
}
```

Find the error in the following function and show how to fix it.

```
int count_periods(const char *filename) {  
    FILE *fp;  
    int n = 0;  
    if ((fp = fopen(filename, "r")) != NULL) {  
        while (fgetc(fp) != EOF)  
            if (fgetc(fp) == '.')  
                n++;  
        fclose(fp);  
    }  
    return n;  
}
```




The function's while loop calls `fgetc` twice, and thus alternates between testing whether end-of-file has been reached or a period has been read.

```
int count_periods(const char *filename) {  
    FILE *fp;  
    int ch, n = 0;  
    if ((fp = fopen(filename, "r")) != NULL) {  
        while ((ch = fgetc(fp)) != EOF)  
            if (ch == '.')  
                n++;  
        fclose(fp);  
    }  
    return n;  
}
```



Write a shell script to list all files that are directories.



```
for eachEntry in *  
do  
    if [ -d "$eachEntry" ]; then  
        echo "$eachEntry"  
    fi  
done
```

Show the output produced by each of the following program fragments. Assume that i, j, and k are int variables.

a) `i = 1;`  
`printf ("%d \", i++ -1);`  
`printf ("%d", i);`

**0 2**

b) `i = 10; j = 5;`  
`printf ("%d \", i++ - ++j);`  
`printf ("%d %d", i, j);`


**4 11 6**

c) `i = 7; j = 8;`  
`printf ("%d \", i++ - --j);`  
`printf ("%d %d", i, j);`

**0 8 7**

d) `i = 3; j = 4; k = 5;`  
`printf ("%d \", i++ - j++ + --k);`  
`printf ("%d %d %d", i, j, k);`

**3 4 5 4**



```
e) i = 5; j = ++i * 3 - 2;  
printf ("%d %d", i, j);
```

**6 16**

```
f) i = 5; j = 3 - 2 * i++;  
printf ("%d %d", i, j);
```

**6 -7**

```
g) i = 7; j = 3 * i-- + 2;  
printf ("%d %d", i, j);
```

**6 23**

```
h) i = 7; j = 3 + --i * 2;  
printf ("%d %d", i, j);
```

**6 15**

What output does the following `for` statement produce?

```
for (i = 5, j = i - 1; i > 0, j > 0; --i, j = i - 1)
    printf ("%d ", i);
```

**5 4 3 2**

Rewrite the following loop so that its body is empty.

```
for (n = 0; m > 0; n++)
```

```
    m /= 2;
```

```
for (n = 0; m > 0; m /= 2, n++)
```

```
    /* empty loop body */
```

Write a program that counts the number of vowels (a, e, i, o and u) in a sentence.

Enter a sentence: And that's the way it is.

Your sentence contains 6 vowels.

```
#include <ctype.h>
#include <stdio.h>
int main(void){
    char ch;
    int num_vowels = 0;
    printf("Enter a sentence: ");
    while ((ch = getchar()) != '\n')
        switch (toupper(ch)) {
            case 'A': case 'E': case 'I': case 'O': case 'U':
                num_vowels++;
        }
    printf("Your sentence contains %d vowels.\n", num_vowels);
    return 0;
}
```

Write a C program to get the Process ID and the Parent Process ID in Linux.

```
#include <stdio.h>
#include <unistd.h>

int main()
{
    int p_id, p_pid;

    p_id = getpid(); /*process id*/
    p_pid = getppid(); /*parent process id*/

    printf("Process ID: %d\n", p_id);
    printf("Parent Process ID: %d\n", p_pid);

    return 0;
}
```

Write a C program to find the size of a file in Linux using `fseek()` and `ftell()` functions. Assume the name of the file is `temp.txt`.

```
#include <stdio.h>
int main(){
    FILE *fp; /*to create file*/
    long int size=0;
    /*Open file in Read Mode*/
    fp = fopen("temp.txt","r");
    /*Move file point at the end of file.*/
    fseek(fp, 0, SEEK_END);
    /*Get the current position of the file pointer.*/
    size = ftell(fp);
    if(size != -1)
        printf ("File size is: %ld\n",size);
    else
        printf ("There is some ERROR.\n");
    return 0;
}
```



Write a program in C to dynamically allocate memory using `malloc()` function to store N integer numbers entered by the user and then print the sum.


```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
    int N = 0, sum = 0, i;
    int *iptr, *tmp;
    printf("Enter value of N [1-10]: ");
    scanf("%d", &N);
    // allocate memory
    iptr = (int *) malloc (N * sizeof(int));
    // check if memory allocated
    if (iptr == NULL) {
        printf("Unable to allocate memory space.\n");
        return -1;
    }
    printf("Enter %d integer number(s)\n", N);
    for (i = 0, tmp = iptr; i < N; i++, tmp++) {
        printf("Enter #%d: ", (i+1));
        scanf("%d", tmp);
        sum += *tmp;
    }
    printf("Sum: %d\n", sum);
    // free memory location
    free(iptr);
    return 0;
}
```

Write a client program in C to connect to the following server code.

```
int main(int argc, char *argv[]) {
    int listenfd = 0, connfd = 0;
    struct sockaddr_in serv_addr;
    char sendBuff[1025];
    time_t ticks;
    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sendBuff, '0', sizeof(sendBuff));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(5000);
    bind(listenfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
    listen(listenfd, 10);
    while(1) {
        connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);
        ticks = time(NULL);
        sprintf(sendBuff, "%.24s\r\n", ctime(&ticks));
        write(connfd, sendBuff, strlen(sendBuff));
        close(connfd);
        sleep(1);
    }
}
```


```
int main(int argc, char *argv[]) {
    int sockfd = 0, n = 0;
    char recvBuff[1024];
    struct sockaddr_in serv_addr;
    memset(recvBuff, '0', sizeof(recvBuff));
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\n Error : Could not create socket \n");
        return 1;
    }
    memset(&serv_addr, '0', sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000);
    if( connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        printf("\n Error : Connect Failed \n");
        return 1;
    }
    while ( (n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0) {
        recvBuff[n] = 0;
        if(fputs(recvBuff, stdout) == EOF) {
            printf("\n Error : Fputs error\n");
        }
    }
    if(n < 0)
        printf("\n Read error \n");
    return 0;
}
```

Which process does an orphan belong to?

- 
- a) root
  - b) parent
  - c) init
  - d) child
  - e) owner


Write a C program implement the following shell command:

ps aux | grep root | grep sbin




```
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <stdio.h>


void exec1();
void exec2();
void exec3();
int pid, pipe1[2], pipe2[2];
```



```
int main() {
    // create pipe1
    if (pipe(pipe1) == -1) {
        perror("bad pipe1");
        exit(1);
    }
    // fork (ps aux)
    if ((pid = fork()) == -1) {
        perror("bad fork1");
        exit(1);
    } else if (pid == 0) {
        // stdin --> ps --> pipe1
        execl();
    }
    // parent
    // create pipe2
    if (pipe(pipe2) == -1) {
        perror("bad pipe2");
        exit(1);
    }
}
```




```
// fork (grep root)
if ((pid = fork()) == -1) {
    perror("bad fork2");
    exit(1);
} else if (pid == 0) {
    // pipe1 --> grep --> pipe2
    exec2();
}
// parent
// close unused fds
close(pipe1[0]);
close(pipe1[1]);
// fork (grep sbin)
if ((pid = fork()) == -1) {
    perror("bad fork3");
    exit(1);
} else if (pid == 0) {
    // pipe2 --> grep --> stdout
    exec3();
}
// parent
return 0;
}
```




```
void execl() {  
    // input from stdin (already done)  
    // output to pipe1  
    dup2(pipe1[1], 1);  
    // close fds  
    close(pipe1[0]);  
    close(pipe1[1]);  
    // exec  
    execlp("ps", "ps", "aux", NULL);  
    // exec didn't work, exit  
    perror("bad exec ps");  
    _exit(1);  
}
```





```
void exec2() {  
    // input from pipe1  
    dup2(pipe1[0], 0);  
    // output to pipe2  
    dup2(pipe2[1], 1);  
    // close fds  
    close(pipe1[0]);  
    close(pipe1[1]);  
    close(pipe2[0]);  
    close(pipe2[1]);  
    // exec  
    execlp("grep", "grep", "root", NULL);  
    // exec didn't work, exit  
    perror("bad exec grep root");  
    _exit(1);  
}
```



```
void exec3() {
    // input from pipe2
    dup2(pipe2[0], 0);
    // output to stdout (already done)
    // close fds
    close(pipe2[0]);
    close(pipe2[1]);
    // exec
    execlp("grep", "grep", "sbin", NULL);
    // exec didn't work, exit
    perror("bad exec grep sbin");
    _exit(1);
}
```




# Midterm Solutions

Let f be the following function:

```
int f(char *s, char *t){
    char *p1, *p2;
    for (p1 = s; *p1; p1++) {
        for (p2 = t; *p2; p2++)
            if (*p1 == *p2) break;
        if (*p2 == '\0') break;
    }
    return p1 - s;
}
```

- a) What is the value of f("abcd", "babc")? **3**
- b) What is the value of f("abcd", "bcd")? **0**
- c) In general, what value does f return when passed two strings s and t?

**The length of the longest prefix of the string s that consists entirely of characters from the string t. Or, equivalently, the position of the first character in s that is not also in t.**



Recall programming assignment 1 (PA1), `count_small <filesize>`.  
Write a line to execute `count_small` with a filesize of 1200 bytes and redirecting the standard input so that the program reads from the `ls` command (with the proper command-line options).

```
$ ls -l | ./count_small 1200
```



Your current working directory has two files prog.c and helper.c. prog.c has a main function in it and calls functions from helper.c.

Write a Makefile with one rule so that when you type make prog, the executable prog will be created only if one or both of the files prog.c and helper.c have been modified. Note that the program should be compiled with the -Wall and -g flags.

```
prog : prog.c helper.c
```

```
gcc -Wall -g -o prog prog.c helper.c
```

What will be the value of the string s1 after the following statements have been executed? Circle the right answer.

```
strcpy (s1, "computer");  
strcpy (s2, "science");  
if (strcmp (s1, s2) < 0)  
    strcat (s1, s2);  
else  
    strcat (s2, s1);  
s1[strlen(s1) - 6] = '\0';
```

- a) co
- b) computerc
- c) computers**
- d) s
- e) computer

Suppose that we call scanf as follows:

```
scanf ("%d%s%d", &i, s, &j);
```

If the user enters 12abc34 56def78, what will be the values of i, s, and j after the call? (assume that i and j are int variables and s is an array of characters). Circle the right answer.


**a) i = 12, s = abc34, j = 56**

b) i = 12, s = abc, j = 34

c) i = 1, s = 2abc34, j = 5

d) i = 1, s = 2abc, j = 34





The following function calls supposedly write a single new-line character, but some are incorrect. Identify which calls don't work and explain why.

a) `printf ("%s", "\n");`

b) `printf ('\n');` **Incorrect; printf requires a string, not a character**

c) `printf ("\n");`

d) `printf ("%c", "\n");` **Incorrect; use "%c" only to write a char value**

Suppose that the following declarations are in effect:

```
int a[] = {5, 15, 34, 54, 14, 2, 52, 72};
```

```
int *p = &a[1], *q = &a[5];
```

a) What is the value of  $*(p+3)$ ? **14**

b) What is the value of  $*(q-3)$ ? **34**

c) What is the value of  $q - p$ ? **4**

What will be the contents of the a array after the following statements are executed?

```
#define N 10
int a[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int *p = &a[0], *q = &a[N - 1], temp;
while (p < q) {
    temp = *p;
    *p++ = *q;
    *q-- = temp;
}
```

**The loop reverses the contents of a. After the statement is executed, a will contain the following data: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.**

If i is an int variable and p and q are pointers to t, which of the following assignments are legal? Circle the right answer.

- a)  $p = i$ ; **illegal because p is a pointer to an integer and i is an integer.**
- b)  $\&p = q$ ; **illegal because  $\&p$  is a pointer to a pointer to an integer and q is a pointer to an integer**
- c)  $p = \&q$ ; **legal**
- d)  $p = *q$ ; **illegal because p is a pointer to an integer and  $*q$  is an integer**
- e)  $*p = *q$ ; **legal**

Consider the following "mystery" function:

```
void pb (int n) {  
    if ( n != 0) {  
        pb (n / 2);  
        putchar ('0' + n % 2);  
    }  
}
```

What does the function do? **Binary representation of n**

Write a program that calls the function, passing it a number entered by the user.

```
#include <stdio.h>

void pb(int n);


int main(void)
{
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    printf("Output of pb: ");
    pb(n);
    printf("\n");
    return 0;
}

void pb(int n) {
    if (n != 0) {
        pb(n / 2);
        putchar('0' + n % 2);
    }
}
```

What will be the output of the following program?

```
#include <stdio.h>
void swap(int a, int b);
int main(void) {
    int i = 1, j = 2;
    swap(i, j);
    printf("i = %d, j = %d\n", i, j);
    return 0;
}
void swap (int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```

**i = 1, j = 2. (The arguments to swap are passed by value, so even though the values of a and b are swapped, the values of i and j are not.)**



One of oldest known encryption techniques is the Caesar cipher. It involves replacing each letter in a message with another letter that is a fixed number of positions later in the alphabet. (If the replacement would go past the letter Z, the cipher "wraps around" to the beginning of the alphabet. For example, if each letter is replaced by the letter two positions after it, then Y would be replaced by A, and Z would be replaced by B.) Write a program that encrypts a message using a Caesar cipher. The user will enter the message to be encrypted and the shift amount (the number of positions by which letters should be shifted):

Enter message to be encryted: Go ahead, make my day.

Enter shift amount (1-25): 3

Encrypted message: Jr dkhdg, pdnh pb gdb.

Notice that the program can decrypt a message if the user enters 26 minus the original key:

Enter message to be encryted: Jr dkhdg, pdnh pb gdb.

Enter shift amount (1-25): 3

Encrypted message: Go ahead, make my day.

You may assume that the message does not exceed 80 characters. Characters other than letters should be left unchanged. Lower-case letters remain lower-case when encrypted, and upper-case letters remain upper-case. Hint: To handle the wrap-around, use the expression  $((ch - 'A') + n) \% 26 + 'A'$  to calculate the encrypted version of an uppercase letter, where *ch* stores the letter and *n* stores the shift amount. (You'll need a similar expression for lowercase letter).



```
#include <stdio.h>
#define MAX_MSG_LEN 80
int main(void){
    char ch, msg[MAX_MSG_LEN];
    int n = 0, i, shift;
    printf("Enter message to be encrypted: ");
    while ((ch = getchar()) != '\n')
        msg[n++] = ch;
    printf("Enter shift amount (1-25): ");
    scanf("%d", &shift);
    for (i = 0; i < n; i++)
        if ('A' <= msg[i] && msg[i] <= 'Z')
            msg[i] = ((msg[i] - 'A') + shift) % 26 + 'A';
        else if ('a' <= msg[i] && msg[i] <= 'z')
            msg[i] = ((msg[i] - 'a') + shift) % 26 + 'a';
    printf("Encrypted message: ");
    for (i = 0; i < n; i++)
        putchar(msg[i]);

    printf("\n");
    return 0;
}
```



Good Luck !