PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

AUGUST 2011 EXAMINATIONS

CSC 209H1Y
Instructor: Daniel Zingaro

Duration — three hours

Examination Aids: one two-sided 8.5x11 sheet.

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

Last (Family) Name(s): _____

First (Given) Name(s): _____

---

*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 examination consists of 10 questions on 16 pages (including this one).

Instructions:

- Check to make sure that you have all 16 pages.
- Read the entire exam before you start.
- Not all questions are of equal value, so budget your time accordingly.
- You do not need to add `import` lines or do error checking
- If you use any space for rough work, indicate clearly what you want marked.

MARKING GUIDE

# 1:	_____/12
# 2:	_____/ 6
# 3:	_____/ 5
# 4:	_____/ 5
# 5:	_____/ 6
# 6:	_____/10
# 7:	_____/ 7
# 8:	_____/ 7
# 9:	_____/12
# 10:	_____/ 1
TOTAL:	_____/71

Good Luck!

CONT'D...

Question 1. [12 MARKS]

Part (a) [2 MARKS]

Write a single Bourne shell command that runs program `foo` with a commandline argument of `bar`, and sends the output to file `outf`.

Part (b) [2 MARKS]

The following shell commands retrieve the number of lines in file `restaurant list` that contain the word `Toronto`. However, all quoting has been removed. Add the proper types of quotes where ever they are required.

```
a=cat restaurant list
b=echo $a | grep -c Toronto
```

Part (c) [2 MARKS]

```
int arraySum (int a[]) {
  ... // don't write code
}
```

Explain why `sizeof(a)` **cannot** be used to obtain the number of elements in the array.

Part (d) [2 MARKS]

How many times does the **original parent process** output `hello\n`?

```
int main(void) {
  printf("hello\n");
  fork();
  fork();
  printf("hello\n");
  return 0;
}
```

Part (e) [2 MARKS]

Briefly explain what happens between the time I hit ctrl-c and the time the process terminates.

Part (f) [2 MARKS]

When a program is executing a handler for signal type `s`, no other signal of type `s` can interrupt the handler. Yet, if a signal of type `s` is received while executing the signal handler, the signal of type `s` will be delivered once the handler completes. Is signal type `s` being **blocked** or **ignored** in the handler? Explain.

Question 2. [6 MARKS]

Consider text files where each line consists of zero or more integers. Here is one sample file:

```
4 8 1
6

5 4 3 2 1
```

Write a Bourne shell program that takes zero or more such files on its commandline, and outputs the total value of adding up all of the integers from all of the lines of all of the files. Your program should output only the total (i.e. a single integer) and nothing else.

Question 3. [5 MARKS]

Consider the following C code.

```
int i = 1;
int *p[5]; //b
int **pp;

p[0] = &i;
pp = &p[0]; //c
printf("%d\n", **pp); //d
```

Part (a) [1 MARK]

What is the type of each element in `p`?

Part (b) [1 MARK]

Write the equivalent of the `//b` line using `malloc` instead of an array.

Part (c) [1 MARK]

Write the equivalent of the `//c` line without using the array `[]` notation.

Part (d) [2 MARKS]

What is the output of the `//d` line? If the line contains an error or tries to access a variable that is not initialized, describe the problem rather than giving the output.

Question 4. [5 MARKS]

Here is part of the man page for a C function called `strspn`.

```
SYNOPSIS
#include <string.h>
size_t strspn(const char *s, const char *accept);

DESCRIPTION
The strspn() function calculates the length of the initial segment of s
which consists entirely of characters in accept.

RETURN VALUE
The strspn() function returns the number of  characters in the initial
segment of s       which consist only of characters from accept.
```

For example, `strspn ("hello", "haeiou")` returns 2: the `h` and `e` are in `accept`, but the `l` is not.

Write an implementation of `strspn`. You may use `strlen`, but do not use any other string functions.

```
size_t mystrspn(const char *s, const char *accept) {
```

Question 5. [6 MARKS]

Write a C function that reads an entire line, including the newline, from a file descriptor `fd`. `fd` is guaranteed to contain a line when `readline` is called. You should read only up to and including the newline character, and no further. Assume that `buf` is large enough to contain any possible line. You must use `read` (not `fread` or `fgets` or anything else) to read from the file descriptor. The function should return `-1` on error and `0` on success. If the function succeeds, `buf` should be a valid string (i.e. with a `'\0'` at the end).

```
int readline(int fd, char *buf) {
```

Question 6. [10 MARKS]

Consider the following `typedef`, which creates a new type called `Pair`.

```
typedef struct node {
  int x, y;
  struct node *next;
} Pair;
```

Part (a) [2 MARKS]

The code below creates several nodes and connects some of them through `next` pointers. Draw the linked list that starts at `a`. Don't draw anything that is not reachable from `a`.

```
Pair *a = malloc(sizeof(Pair));
Pair *b = malloc(sizeof(Pair));
Pair *c = malloc(sizeof(Pair));

a->x = 1;
a->y = 2;
b->x = 3;
b->y = 4;
c->x = 5;
c->y = 6;
a->next = b;
a->next->next = c;
b->next = NULL;
```

Part (b) [3 MARKS]

`pl` is a linked list of `Pair`s. Write a function `isxVal` that returns a true value if `val` exists as any `x`-component of a `Pair` in `pl`, and a false value otherwise.

```
int isxVal (const Pair *pl, int val) {
```

Part (c) [5 MARKS]

Write a function `copyReverse` that creates a new linked list consisting of all nodes in `old`, but in reverse order. The new list and the old should be independent, so that modifying an `x` or `y` in one list does not modify the other list. The function returns the head of the new list using the second parameter of the function (not through a `return` statement!).

```
void copyReverse (Pair *old, Pair **new) {
```

Question 7. [7 MARKS]

Part (a) [4 MARKS]

When I run the following code, one of two things happens:

- Scenario I: There is no output at all, or
- Scenario II: The child prints `Wrote in pipe\n`

```
int main(void) {
  int child;
  int fd[2];

  if (pipe(fd) == -1) {
    perror ("pipe");
    exit (1);
  }
  if ((child = fork()) == -1) {
    perror ("fork");
    exit (1);
  } else if (child == 0) {
    char *s = "Lake Hylia\n";
    close (fd[0]);
    write (fd[1], s, strlen(s));
    printf ("Wrote in pipe\n");
  } else {
    close (fd[1]);
    /*Code for part b goes here*/




    close (fd[0]);
    exit (0);
  }
  return 0;
}
```

The reason that there are two scenarios has to do with the order that things happen after the `fork`. Carefully describe what happens after the `fork` to cause each of these two scenarios.

Scenario I:

Scenario II:

Part (b) [3 MARKS]

Fill in the missing parent code so that it reads the string from the pipe and prints it on standard output. **Do not** hard-code the number `11`.

Question 8. [7 MARKS]

Write a function that reads from `fromfd` and copies what it reads to `tofd`. `fromfd` is a file descriptor open for reading and `tofd` is a file descriptor open for writing. The function continues until one of two things happens:

- There is an end-of-file on `fromfd`, or
- There is read action (i.e. any bytes to read, or end-of-file) on `quitfd`. `quitfd` is a file descriptor open for reading.

You must use `select` in your solution: you are not allowed to block on one file descriptor when there could be action on the other.

```
void fdecho (int fromfd, int tofd, int quitfd) {
```

Question 9. [12 MARKS]

Part (a) [2 MARKS]

Explain what is meant by the **Network Newline Convention**. Why is such a convention necessary?

Part (b) [2 MARKS]

A client would like to write an integer to a socket. Which function, `htons` or `ntohs`, should first be called on that integer? Explain why calling that function is necessary.

Part (c) [8 MARKS]

In this part, you will write the body of a small server. Assume that all setup has been done and that you have a global socket file descriptor `listenfd` that is in the listening state.

Your server will accept one client at a time. When each client `c` connects:

- Send `c` all of the text from the previous client (if `c` is the first client, nothing gets sent to them), and then
- Read from `c` until `c` disconnects or your buffer is full. If your buffer fills, empty the buffer and disconnect the client.

For example, when the first client connects, you send them nothing and read from them until they drop or your buffer gets full. Let's say that you read the text `rupee` from them. Then, later, when the second client connects, you send them `rupee` and read from them until they disconnect. If you read the text `lanmolas` from them, you send `lanmolas` to client 3 ... and so on. If one of the clients sends you too much text (so that your buffer is overflowed), the client after them does not get any text from you.

```
//Here is the syntax for accept
//accept(listenfd, (struct sockaddr *)&r, &socklen)

int listenfd;

int main(void) {
  struct sockaddr_in r;
  socklen_t socklen = sizeof (r);
  setup();
  //socket(), bind(), listen() done; listenfd is listening
  //Declare all variables and write code
```

Question 10. [1 MARK]

Sometimes, we learn important things besides course content. In this course, you learned (circle one):

- You are still convinced that it is me singing `love.wav`
- You are rocking `cartoon.wav` on your MP3 player
- You are going to play Zelda 3 for the rest of the summer and then talk to me about its epicness
- You think that `malloc` should be a four-letter word
- Tiles are supposed to do that
- You never want to see another `glibc: corruption or double free` error again
- You suddenly love Western Meadowlarks

It's been a privilege. Have a great summer!

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*