COMP 206 Fall 2018 – Midterm Study Guide

Intro:

This document is meant to assist your studying. It's not guaranteed to be exhaustive and you must cover the material given in lectures even if it is not represented here. The format of this document is not identical to the midterm. I do guarantee that going through these topics and questions will be helpful for your midterm preparation, but it should not be the only studying that you do.

Material Covered:

- All lectures from the start of term until Friday October 12th, 2018
- Slides up to "Lecture12-Memory"
- K&R text Chapters 1-5 and 7
- Incomplete list of definitions and concepts:
    - Filesystem
    - Command-line arguments
    - Standard input/output
    - Wildcards
    - Interpreted vs compiled languages
    - Pass-by-value
    - C strings
    - Pointers and Arrays
    - File IO
    - 2D arrays
    - Arrays of pointers
    - Memory regions within a running process: stack, heap, data, text, BSS (not needed: kernel or mem map)
- Incomplete list of programming elements in BASH:
    - Frequently used commands
    - Variables
    - For loops
    - While loops
    - If conditionals
    - Command substitution to variable $() or ``
    - Pipes
    - Input and output re-direction
    - Performing math
    - Wildards
    - Quoting
- Incomplete list of programming elemnts in C:
    - Variables

- Basic control constructs: while, for, if
- Data types
  - Characters
    - AASCI table
    - Equivalent integer/binary representations
  - Integers:
    - Binary values
    - Various “lengths” and their memory implications
  - Floats:
    - Conversion to/from integers
- Functions:
  - Return values
  - Pass by value
  - Variable scope
- Text data
  - Printing to standard output or a file
  - Reading from standard input or a file
  - Parsing text, lines into words, words into characters, based on delimiters such as comma, etc
  - Creating text, building lines from words, words from characters, CSV, etc
  - Concept of \0, use in various functions such as string length
- Arrays and pointers:
  - How they are similar and different
  - Dereference operator
  - Array indexing
  - Relation between the two above
  - Working properly with array lengths
  - Using pointers to emulate "pass by reference”
  - 2D arrays
  - Arrays of pointers
- Memory:
  - Parts of a running process
  - Stack memory limitations
  - Malloc and the heap

Short Questions

The official midterm will have 8 multiple choice questions. Some will be similar to Quiz questions you have seen and others will be slightly more involved, such as the list given here. The midterm will be done completely on the scantron (bubble sheets), but here we have a variety just to help you think about the concepts.

Assume that the file named abcABC123.c exists in the current folder. Circle all of the following commands that result in its name being printed:

a) ls [abcdef]
b) ls *CBA*
c) ls *[abcdef]*
d) ls *[hijklm]*
e) echo *

Circle and number at least 3 different bugs/errors in either logic or syntax in the following code fragment. Provide a written explanation for each one below.

```
File f;
char *name="test.dat";
char *name2="test2.dat";
char a=getchar(); // User enters F or S to specify file
char *buffer;
if ( a = "F" ) {
   f=fopen(name);
}
else{
   f=fopen(name2);
}
fgets(buffer,1024,f);
```

In the C language, what value does the variable x take on?

```
int main() {
   int x;
   int y;
   y = 'a' – 'b';
   x  =  y++  * 2;
}
```

Circle at least 3 problems in the following C program. For each, briefly describe the problem, and suggest improved code:

```
#include <stdio.h>

int TheMain( int argc, char* argv[] ){
        char *s = argv[1];

        while( *s ){
        char *rest_of_string = s.substr(1,end);

        if( ++(*s) == \0 )
        System.out.printf( “End of string.\n” );
        break;
        }

        return s[1];
}
```

What does this C program print on the terminal?

```
        #include <stdio.h>

        void muddle( char string[] ){

              string[0] = 'M' + 1;
              string[1] = string[1];
              string[2] = string[1];

              char temp = string[4];
              string[4] = string[5];
              string[5] = temp;
        }

        int main()
        {
              char string[10] = "goedel";
              muddle( string );
              printf("%s\n", string );
              return 0;
        }
```

What does the following program print?

```
#include <stdio.h>
int swap( int *i, int *j ){
        int temp = *i;
        *i = *j;
        *j = temp;
}
int main()
{
        int a=2;
        int b=3;
        int *temp = &a;
        swap( temp, &b );
        printf("%d %d %d\n",a, b, *temp);
        return 0;
}
```

a) 2 3 2
b) 3 2 3
c) 2 2 2
d) 3 3 2
e) None of these choices

What does the following C program output to the terminal when compiled and run?

```
#include <stdio.h>

int main(){
        char *my_word_array[4] = { NULL, NULL, NULL, NULL };

        char first[100] = “one”;
        my_word_array[0] = first;

        my_word_array[1] = “two”;

        char third[100];
        third[0] = ‘3’;
        my_word_array[2] = third;

        strcpy( my_word_array[3], “4” );

        printf( “The words are: %s %s %s %s.\n”,
                my_word_array[0], my_word_array[1],
                my_word_array[2], my_word_array[3] );
}
```

a) The words are: one two 3 4.
b) The words are: first two third 4.
c) The output will be “The words are: one two 3” followed by up to 195 random characters and a period.
d) Segmentation fault
e) The output will be “The words are: one” followed by up to 97 random characters, “two” followed by up to 97 random characters, “3” followed by up to 99 random characters, followed by up to 100 random characters, followed by a period.

Consider the following C program. Which choice best describes its outcome:

```
#include <stdio.h>

void main()
{
        int i=2;
        int x=3;
        while( 1 ){
                x=x+x;
                break;
        }
        for ( i=i+i ; i<x ; x++ ){
                i=i+i;
        }
        printf("%d %d\n",x,i);
}
```

a) Displays “9 4” to terminal
b) The program produces an infinite loop
c) Displays “7 8” to the terminal
d) The program does not compile
e) Displays “4 6” to terminal

Written Questions

The official midterm will have 2 written programming questions. They will typically take about ¼ to ¾ of a page to complete. They can be small pieces of programs where a portion is provided or can require you to write a simple program entirely.

Simple calculator. Produce a C program that can be run with an alternating series of integer numbers and math operators (plus, minus, multiply, divide = +, -, x, /), provided as command-line arguments. Your program must perform the math operations requested. **Do not respect "order of operations"** but rather simply apply the operations left to right.

No argument checking is needed. Assume your program is only ever run with correctly formatted arguments and that the order is always correct as: number operator number operator... Assume all number and the result will fit in a 32-bit integer and don't do anything special to worry about integer division and round-off (just apply the right C math operations).

Examples:

$ ./a.out 3 + 4

7

$ ./a.out 3 - 4 x 2

-2

$ ./a.out 8 / 4 x 2 - 5 + 10

9

Write a C program takes a single command-line argument, the name of a file. The program reads this file, and outputs on standard out only the first letter of each word (treat only spaces or newlines as valid ways to separate words).

Create a function with the following signature:
void encrypt(char source[], char destination[], int key);
That performs the following action: The characters in source are encrypted into destination using the integer key. Minimise the use of library calls, none if possible. Encryption will be based on Reverse Caesar Cipher, which shifts a character by a positive integer number and stores the characters in the destination array in reverse order. If the character is A and the integer number is 2, then the encrypted character is C. If the input character is Y and the integer number is 3 then the encrypted character is B (note shift means with wrap-around). If the input characters are ABC with the key 2 then the output array would contain EDC, the output shifted by 2 in reverse order.