McGill

Practice Midterm #2

VERSION A

INTRODUCTION TO SOFTWARE SYSTEMS

COMP 206 – Section 707

INSTRUCTIONS:

- Write your answers on the provided multiple choice form and the final two pages of this booklet only. Any other markings on the earlier pages of this test will not be marked.
- Make sure to include your name and student number on the top of each programming page.
- This is a **CLOSED BOOK** examination.
- Only one double-sided **CRIB SHEET** permitted.
- This is partially a **MULTIPLE CHOICE** examination.
    - The Examination Security Monitor Program detects pairs of students with unusually similar answer patterns on multiple-choice exams. Data generated by this program can be used as admissible evidence, either to initiate or corroborate an investigation or a charge of cheating under Section 16 of the Code of Student Conduct and Disciplinary Procedures.
- You are permitted **TRANSLATION** dictionaries ONLY.
- **NO CALCULATOR** is permitted.
- This examination is **PRINTED ON BOTH SIDES** of the paper
- This examination paper **MUST BE RETURNED**
- This examination consists of 10 multiple choice questions and two programming questions. The document should be 10 pages, including the cover page.
- There are 60 marks overall and 60 minutes to write. Consider using the mark value of questions to budget your time wisely, 1 minute per mark.

Section 1: Multiple Choice Questions

Question 1:

For the BMP files that we've seen in class, how can one tell the size of the header?

a) It is specified by the header field ***width***
b) It is a constant ***size of*** 54 bytes
c) It is returned by the ***fread()*** function on the file
d) It is specified by the header field ***offset to the pixel data***
e) It is indicated by the position of the ***first comma***

Question 2:

Considering the specification of a structure:

```
typedef struct s_student{
  int mcgill_id;
  char name[100];
} STUDENT;
```

Which of these is a valid method to create a variable of the structure:

a) struct STUDENT s;
b) typedef s_studdent s;
c) struct s_student STUDENT;
d) STUDENT struct s_student;
e) STUDENT s;

Question 3: Which of the following are true comparisons of BUFFERED vs UNBUFFERED IO:

a) BUFFERED IO is a more efficient way to read large files from disk
b) UNBUFFERED IO is the default provided in <stdio.h>
c) UNBUFFERED IO cannot be used when interacting with the user via keyboard
d) There is no difference between BUFFERED and UNBUFFERED IO
e) BUFFERED IO needs to be used when reading binary data files, with the 'b' flag in fopen

Question 4: In C, one can access and use the address of which of the following:

a) Only local variables
b) Local and global variables only
c) Any variables, and the local functions in your own code
d) All variables, and functions with type void
e) All variable and all functions

Question 5: Which is a true statement regarding STATIC vs DYNAMIC libraries produced on Linux by gcc:

a) STATIC libraries are used to create small, efficient executables
b) DYNAMIC libraries produce executables with fewer run-time dependencies
c) The LD_LIBRARY_PATH variable is useful for finding DYNAMIC libraries
d) STATIC libraries can only hold code from one single .c file
e) DYNAMIC libraries can only hold code from one single .o file

Question 6: What is the outcome of the following C program?

```
#include <stdio.h>

struct first_struct{
  int i;
  char c;
}

void main(){
      struct first_struct a = { 1, 'a' };
      struct first_struct b = { 2, 'b' };

      a.i = b.i;
      b.c = a.c;

      a.i++;
      b.c++;

      printf ("%d %c %d %c.\n", a.i, a.c, b.i, b.c );
}
```

a) It does not compile
b) It fails to run with a segfault
c) It displays 3 b 3 b to the terminal
d) It displays 3 a 2 b to the terminal
e) It displays 1 a 2 b to the terminal

Question 7: What does the following C program display to the terminal?

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

#define SEASON "winter"

void main(){

   char puddle[100] = "None.";

   #ifdef SPRING
      strcpy( puddle, "Large" );
   #endif

   if( strcmp( SEASON, "winter" ) == 0 )
      strcpy( puddle, "Potholes." );

    printf( "The season is %s and the puddles are %s.\n", SEASON, puddle );
}
```

a) The season is SPRING and the puddles are Large.
b) Segfault
c) The season is winter and the puddles are Potholes.
d) The season is SEASON and the puddles are None.
e) The season is “winter” and the puddles are Potholes.

Question 8: After declaring an array of struct data in this fashion:

```
A:    struct s_image{
B:      int width;
C:      int height;
D:      unsigned char* pixel_data;
E:    };
F:
G:    struct s_image image_array[100];
```

Which of the following lines would produce compilation errors or guaranteed run time crashes? In this case assume each line is run independently, as the first line of the main() function.

```
1:    image_array[0].width = 100;
2:    (*(image_array+5)).height=100;
3:    image_array->pixel_data = (unsigned char*)malloc(100);
4:    struct s_image s = { 100, 100, "BMP" };
5:    (image_array+50)->height = 100;
6:    *(int*)(image_array+50) = 50;
7:    strcpy( (char*)(image_array+2*sizeof(int)), "BMP" );
```

a) All lines cause an error
b) 3, 4, 6, 7
c) 4 and 6
d) 6 and 7
e) No lines cause an error

Question 9: Consider a project with 3 .c and 2.h files:

```
sort_main.c            // Holds the program's main function, includes both swap.h and min.h
min.h and min.c        // Include the min() function's declaration and implementation
swap.h and swap.c      // Include the swap() function's declaration and implementation
```

We want to build the sort_main program with the single command-line:

```
$ make
```

The Makefile starts as follows:

```
OBJS=min.o swap.o
HEADERS=min.h swap.h
SRC=sort_main.c min.h min.c

all: sort_main

sort_main: <missing section A>
	<missing section B>

%.c.o:
	gcc –c $<
```

What are the correct completions for sections A AND B listed?

a) ${OBJS} AND gcc ${OBJS} sort_main.c –o sort_main
b) ${OBJS} ${HEADERS} AND gcc –c sort_main.c –o sort_main.o
c) sort_main.c AND gcc sort_main.c min.h swap.h –I sort_main
d) ${SRC} AND gcc –c ${OBJS} sort_main.c –L sort_main
e) sort_main.o AND gcc sort_main.o –o sort_main

Question 10: What is the outcome of the following program?

```
#include <stdio.h>

int add( int a, int b ){  return a+b;  }
int mult( int a, int b ){  return a*b;  }

int main(){

        int(*first)(int,int) = &add;
        int(*second)(int,int) = first;

        first = &mult;
        printf( "Result: %d.\n", first( 3, second(5, 7) ) );
        return 0;
}
```

a) Compilation error
b) Segfault
c) Prints Result: 15.
d) Prints Result: 105.
e) Prints Result: 36.

Section 2: Short coding questions:

You must answer the next 2 questions directly on this exam. Ensure to write your name and McGill ID on top of each page.

Question 11: (15 marks) Factor the single C file, full_prog.c on this page into 3 files on the following page: main.c, fncs.c and fncs.h such that the only function in main.c is main(), but the program can still be compiled correctly and have the identical functionality. The 3 files will be compiled together with:

```
$ gcc main.c fncs.c
```

——— CONTENTS OF full_prog.c —————

```
#include <stdio.h>
#define FIRST 5

int global_a = FIRST;

int fn1( int a ){ return a+5; }
int fn2( int b ){ return global_a + b;

int main(){
        int c = FIRST;
        global_a += fn1( c );
        printf( "%d", fn2( c ) );
        return 0;
}
```

——— END full_prog.c ————————

———— CONTENTS OF fncs.h (you complete) ————

———— END fncs.h ————

———— CONTENTS OF fncs.c (you complete) ————

———— END OF fncs.c ————

———— CONTENTS OF main.c (you complete) ————

———— END OF main.c ————

Question 12: (15 marks) Complete the C insert_one_sorted function that sorts an array of numbers, given as a global variable input_array, into a C linked list following the structure specified by LIST_NODE. Ideal program output: 0123456789

```
#include <stdio.h>
#include <stdlib.h>
int input_array[10] = { 3, 7, 1, 8, 2, 4, 9, 5, 6, 0 };
typedef struct s_list_node{
        int val;
        struct s_list_node *next;
} LIST_NODE;

LIST_NODE* insert_one_sorted( LIST_NODE* list_so_far, int val_to_sort ){
    // YOUR SOLUTION FROM HERE

















    // TO HERE!
}

int main(){
        LIST_NODE *head = NULL;
        for( int i=0; i<10; i++ ) head = insert_one_sorted( head, input_array[i] );
        while(*head!=NULL)      {
                printf( “%d”, head->val );
                head = head->next;
        }
        return 0;
}
```