Spring Semester 2015

KAIST EE209

Programming Structures for Electrical Engineering

Mid-term Exam

Name: ______________________________

Student ID: ______________________________

This exam is closed book and notes. Read the questions carefully and focus your answers on what has been asked. You are allowed to ask the instructor/TAs for help only in understanding the questions, in case you find then not completely clear. Be concise and precise in your answers and state clearly any assumption you may have made. You have 135 minutes to complete your exam. Be wise in managing your time. Good luck.

Question 1	/ 10
Question 2	/ 10
Question 3	/ 20
Question 4	/ 10
Question 5	/ 10
Question 6	/ 20
Total	/ 80

Name: Student ID:

1. Basic concepts (10 points)

Please answer the following two questions based on the supplemental material uploaded in KLMS.

(1) Compilation Steps

Suppose that you have the C source file "hello.c". Making an executable file, say "hello" from hello.c requires the compilation system to take 4 phases. Explain those 4 phases, i.e., what each phase does, who performs each phase etc. (5 points)

Name: Student ID:

(2) Hardware organization of a typical computer system

Draw a diagram of the hardware organization of a typical computer system and briefly explain the role of each hardware component. (5 points)

Name: Student ID:

2. Multiple source files which generate one executable program (10 points)

You are in a big software project, and 3 programmers are involved in this project. You are response for making a module related to controlling a laser printer. You plan to implement this module in the source file “printer.c”, where you will implement the following two functions (which will be called by the modules of other two programmers):

```
(1)int * print_document(const char * str)
(2)char * check_availability(char * printer_driver)
```

You also have a global variable called “**`int errno`**”, which stores some printing errors. Write the necessary codes in each of “printer.h” and “printer.c”.

(Note that “printer.h” can be included by other multiple “.c” files, and also the global variable **`errno`** can be used in other “.c” files. You don’t have to write the detailed codes of the above two functions, where just “…..” is enough.)

printer.h

Name: Student ID:

printer.c

Name: Student ID:

3. Pointer (20 points)

(1) Multi-dimensional array and pointer. Here's a loop that clears column **i** of the array **a**. Please declare a variable **p** inside the box, so that the entire codes should be correct (which type should be used for the variable **p**?). (5 points)

```
#define     NUM_ROWS    10
#define     NUM_COLS    20


int   a[NUM_ROWS][NUM_COLS];
int   i;





   ……

i = 3; /* select a column to be cleared */

   ……


for (p = &a[0]; p < &a[NUM_ROWS]; p++)
{
      (*p)[i] = 0;
}
```

Name: Student ID:

(2) What will be the contents of the array **a** after the following statements are executed? (5 points)

```
#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;
}
```

Name: Student ID:

(3) Write the following function:

```
void find_two_largest(const int *a, int n,
                      int *largest, int *second_largest);
```

`a` points to an array of length **`n`**. The function searches the array for its largest and second-largest elements, storing them in the variables pointed by **`largest`** and **`second_largest`**, respectively. Use pointer arithmetic---not array subscripting---to visit array elements. (10 points)

Name: Student ID:

Name: Student ID:

4. Implementing a string function (10 points)

[String] Implement your own strduplicate considering following description.

- **`char * strduplicate (const char * src)`** duplicates the C string pointed by **`src`**, and returns the pointer to the duplicated string (thus, it is safe to assume that the string **`src`** ends with the null character). The duplicated string should be stored in the different memory space from **`src`**.
- **`strduplicate`** returns the pointer to the duplicated string or **`NULL`** if failed.
- You cannot use any string-related function

Following code is example of using **`strduplicate`**. Fill out the blank below.

```
#include<stdio.h>

char * strduplicate(const char* src)
{





}

int main(int argc, char* argv[])
{
  char str [] = "Prof. Yung Yi is very handsome";
  char *dup_str;

  dup_str = strduplicate(str);
  printf("str: %s\ndup_str: %s\n", str, dup_str);

  return 0;
}
```

Output:

```
str: Prof. Yung Yi is very handsome
dup_str: Prof. Yung Yi is very handsome
```

Name: Student ID:

5. Deterministic finite state automaton (DFA) and Regular expression (10 points)

a) See the following character strings and determine each of them matches with the regular expression '^a*bc' or not. (5 points)

(1) bc	(O / X)
(2) abc	(O / X)
(3) bbc	(O / X)
(4) abca	(O / X)
(5) babbc	(O / X)

b) Draw a DFA of the regular expression '^a*bc'. If a state does not specify a particular input character, we assume that the DFA automatically fails on that input character. Assume that 'a', 'b', and 'c' are the only input characters. Your DFA should be the minimal form. (5 points)

Name: Student ID:

6. Structure (20 points)

Write the codes that satisfy the following requirements.

(1) Define **`Student`** structure with members, **`name`**, **`id`**, and **`sex`**, whose types are C string, integer, and *a new type* allowing only MALE and FEMALE.

(2) *Dynamically* allocate the memory that can store three students (static array allocation is not allowed)

(3) Set the allocated structure variables with the following three students:

```
{“yiyung”,3486,MALE}, {“hjlee”,5094,FEMALE}, {“jhlee”,3406,MALE}
```

(4) Print out three students.

Name: Student ID: