Fall Semester 2013

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. All your answers must be included in the attached sheets. You have 120 minutes to complete your exam. Be wise in managing your time. Good luck.

Scores

Question 1	/ 20
Question 2	/ 30
Question 3	/ 15
Question 4	/ 35
Total	/

Name: Student ID:

1. Briefly describe following words. (20 points)

(a) Redirection (2 points)

(b) DFA (2 points)

(c) Ones' and Two's Complement (4 points)

(d) Operators (2 points)

Name: Student ID:

(e) Call by value and reference (4 points)

(f) Recursive function (2 points)

(g) Memory leak (2 points)

(h) Dangling pointer (2 points)

Name: Student ID:

2. Read following C statements and describe the printed output. Assume that we are on a 32-bit machine. (30 points)

(a) (15 points)

```
#include <stdio.h>

int main(void) {

      int i;

      for (i = 3; (i -= 2) >= 0; i++)

            printf("%d\n", i << 2 + 1);

      return 0;

}
```

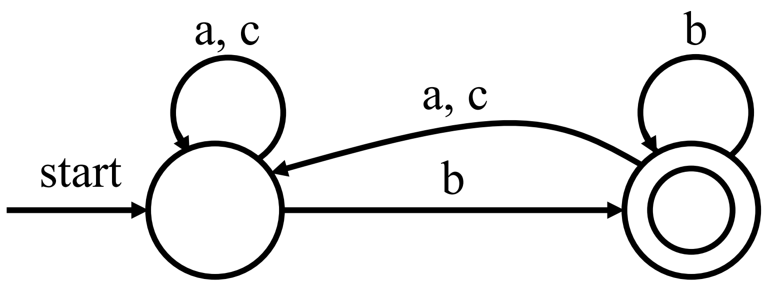
(b) (15 points)

```
#include <stdio.h>

void my_function(int a, int b) {

      int temp = a;

      *(&a) = b;

      *(&b) = temp;

}

int main(int argc, char *argv[]) {

      int x = 3, y, *p, *q = &x;

      p = &y;

      *p = ++(*q);

      my_function(*q, x++);

      printf("x = %d\n", x);

      printf("y = %d\n", y);

      return 0;

}
```

Name: Student ID:

3. See the given regular expressions or DFA, and convert them to corresponding DFA or regular expressions respectively. If a state does not specify a particular input character, we assume the DFA automatically fails on that input character. As in the assignment #2, * matches zero or more occurrence of the previous character while + matches one or more occurrences of the previous character. . matches one of ‘a’, ‘b’, or ‘c’. ^ is the start of the character string while $ is the end of character string. (15 points)

(a) (5 points)



(b) ^a+$ (5 points)

Name: Student ID:

(c) ^.+bc$ (5 points)

Name: Student ID:

4. Strings.

(a) What is the difference between string literal and string variable? (5 points)

(b) Read following C statements. What will be the output printed out? (15 points)

```
#include <stdio.h>
int main(void)
      char cArr[] = “Hello”;
      char *cp = “World”;
      char **ptr;
      int cnt;

      for (cnt = 0, ptr = &cp ; **ptr != *(cArr +cnt) ; cnt++)
            ++(*ptr);

      printf(“%d\n”, cnt);
      printf(“%s\n”, cp);
      return 0;
}
```

Name: Student ID:

(c) Implement your own strncmp considering following description. (15 points)

`int strncmp (const char *str1, const char *str2, size_t num)` compares up to *`num`* characters of the C string *`str1`* to tthose of the C string *`str2`*. This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ, until a terminating null-character is reached, or until *`num`* characters match in both strings, whichever happens first.

It returns an integral value indicating the relationship between the strings: a zero value indicates that the characters compared in both strings form the same string. A value greater than zero indicates that the first character that does not match has a greater value in *`str1`* that *`str2`*; and a value less than zero indicates the opposite.