Fall 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 140 minutes (9:00 AM – 11:20 AM) to complete your exam. Be wise in managing your time. Good luck.

Question 1	/ 10
Question 2	/ 15
Question 3	/ 10
Question 4	/ 10
Question 5	/ 10
Question 6	/ 15
Question 7	/ 15
Question 8	/ 15
Question 9	/ 10
Total	/ 110

Name: Student ID:

1. (10 point) For the arithmetic operation 75 - 91, translate the decimal numbers to 2's complement form, complete the arithmetic operation, and translate the result back into decimal. Assume an 8-bit word size.

Name: Student ID:

2. (15 point) Translate each of the following English-language descriptions into a C declaration.

(a) (5 point) Structure person has three components: name (a pointer to a structure that contains two components fname (a character string), and lname (a character string)); dob (an array of 3 integers), and parent (a pointer to a person).

(b) (5 point) Declare variable **`employees`** as an array of 10 pointers to structure person.

Name: Student ID:

(c) (5 point) Suppose variable **`employees`** is a pointer to a person structure, and variable **`numemps`** is an integer. Show the code to dynamically setup employees (i.e., dynamic memory allocation) as an array of size numemps, with everything allocated initialized to zero.

Name: Student ID:

3. (10 point) What is the output of the following program?

```
#include <stdio.h>

int main(void) {
    char *array[3][3]=
           {   {":|(", "Hope", "you"},
               {"have", "a", "good"},
               {"fall", "break",":|}"}
           };
    char *(*p)[3]=array;
    char **q=*(array+1);

    printf("1 : %s\n",array[0][2]);
    printf("2 : %s\n",*(array[1]));
    printf("3 : %s\n",*(*(array+1)+1));
    printf("4 : %s\n",*(q+2));
    printf("5 : %s\n",*(*(p+2)+1));

    return 0;
}
```

Name: Student ID:

4. (10 point) What string does the function **`f()`** below return? Explain your answer.

```
char* f(unsigned int n){
   int i, numbits = sizeof(unsigned int) * 8;

   char* ret = (char *) malloc(numbits + 1);

   for (i=numbits-1; i>=0; i--, n>>=1)
      ret[i] = '0' + (n & 1);

   ret[numbits] = '\0';

   return ret;
}
```

Name: Student ID:

5. (10 point) What does the program output when executed?

```
int main()
{
    int a = 14;
    int b = 5;
    int c = 29;
    unsigned short x = 0;
    x = a << 12;
    x |= (b & 0x007) << 8;
    x |= c & 0xff;

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

return 0;
}
```

Name: Student ID:

6. (15 point) For each of the following code fragments, determine whether or not it is likely to cause a run-time program error (assume that the code compiles without warnings or errors). If there is a run-time program error, please write "ERROR" next to it and describe the problem in one or two short sentences. Otherwise, simply write "OK" next to it.

a) (5 point)

```
void *pVoid = malloc(8 * sizeof(int));
int *pInt = pVoid;
free(pInt);
```

b) (5 point)

```
float a[] = { 1.0, 0.0, 1.0, 0.0 };
float *b = &a[2];
float *c = b-- + 1;
float result = *b / *c;
```

Name: Student ID:

c) (5 point)

```
void *my_alloc(void *ptr, int size)
{
    if (ptr)
       return realloc(ptr, size);
    else
       return malloc(size);
}

void my_free(void *ptr)
{
    free(ptr);
    ptr = 0;
}

int main()
{
    void *ptr = my_alloc(0, 4);
    my_free(ptr);
    ptr = my_alloc(ptr, 4);
    my_free(ptr);
    return 0;
}
```

Name: Student ID:

7. (15 point) Consider the following function that converts an integer to a string, where the **`sprintf()`** function "prints" to a formatted string, e.g., **`sprintf(retbuf, "%d", 72)`** places the string **`"72"`** starting at the location in memory indicated by the address **`retbuf`**:

```
char *itoa(int n) {
   char retbuf[5];
   sprintf(retbuf, "%d", n);
   return retbuf;
}
```

(a) (5 point) Identify two serious bugs in this function.

(b) (10 point) Rewrite the function to fix these bugs.

8. (15 point) The program shown on the page 13 should output the following:

```
This is a fall00 midterm question.
his is a fall00 midterm question.T
is is a fall00 midterm question.Th
s is a fall00 midterm question.Thi
is a fall00 midterm question.This
is a fall00 midterm question.This
s a fall00 midterm question.This i
a fall00 midterm question.This is
a fall00 midterm question.This is
fall00 midterm question.This is a
```

However, the program has the memory-related bug (it may not have any compile error, but it may generate segmentation fault or generates core dumps).

(a) (10 point) Identify and fix the bug.

(b) (5 point) After fixing the bug, explain what problem might still arise if printMatrixis provided as a library function to be used in big programs. Provide a very short answer.

Name: Student ID:

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

typedef char * CHARBUF;

/* this function prints a 10-line matrix based on the content of
* argument a. If strlen(a)<10, it will print an error message.
*/

void printMatrix(CHARBUF a)
{
    CHARBUF b[10];
    int i;
    char *to_ptr, *from_ptr;

    if (strlen(a) < 10) {
        fprintf(stderr, "string length must be larger than 10\n");
        return;
    }

    for (i=0;i<10;i++) {
        b[i] = (CHARBUF)malloc(sizeof(char) * (sizeof(a)+1));
        if (b[i]==NULL) {
            fprintf(stderr, "no more memory available.\n");
            return;
        }
        for (to_ptr = b[i], from_ptr = a+i; (*from_ptr) != 0; )
            *to_ptr++ = *from_ptr++;
        for (from_ptr = a; from_ptr != a+i; )
            *to_ptr++ = *from_ptr++;
        *to_ptr = 0;
    }

    for (i=0;i<10;i++)
        printf("%s\n",b[i]);
}

void main(void)
{
    printMatrix("This is a fall00 midterm question.");
}
```

Name: Student ID:

9. An undergraduate student wrote a function to print out characters in string in the reverse order. (e.g., given "abc" the program needs to print out "cba".) She wrote the function in three lines of code. Fill in the blanks using two statements including **`printf()`**. For example to print a character you may use **`printf("%c", ch)`**, where **`ch`** is of type char. You cannot use any other standard library function.

   Note she did not use any loop in her program.

```
void print_reverse(const char* str)
{
      if (*str!='\0') {

      _____________________________________________________;

      _____________________________________________________;

     }
}
```