Fall Semester 2013

KAIST EE209

Programming Structures for Electrical Engineering

Final 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	/ 10
Question 2	/ 15
Question 3	/ 15
Question 4	/ 20
Question 5	/ 20
Total	/ 80

Name: Student ID:

1. Briefly describe following words. (10 points)

(a) Context Switch (2 point)

(b) Virtual Memory (2 point)

(c) Temporal Locality and Spatial Locality (2 point)

(d) Internal Fragmentation (2 point)

(e) Critical Section (2 point)

Name: Student ID:

2. Bug Hunting. (15 points)

The following program (in the next page) is meant to find *anagrams*. For each word (such as "beaters") it sorts the letters into alphabetical order (such as "abeerst") and enters the pair in a table. Since anagrams (such as "beaters" and "berates") have the same key ("abeerst"), one can find anagrams by looking up in the table. (don't care about the memory and the input sizes.)

For an input file such as,

```
abacus
abbess
accent
alert
alexia
alter
beaters
beauty
berates
```

the output of the program is supposed to be,

```
alter is an anagram of alert
berates is an anagram of beaters
```

However, there are 7 bugs in the program. Correct the bugs.

(Note: gcc doesn't notice any problem with this program; it's syntactically correct.)

Name: Student ID:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define TABLESIZE 10000

void swap(char a, char b) {
  char t = a;
  a = b; b = t;
}

char *sort(char *s) {
  char *p; int i,j;
  int n = strlen(s);
  for (i=0; i<n; i++)
    p[i]=s[i];

  for (i=0; i<n; i++)      /* no bugs in this line */
    for (j=0; j<i; j++)    /* no bugs in this line */
      if (p[j]>p[j+1])     /* no bugs in this line */
        swap(p[j], p[j+1]);
  return p;
}

struct entry {char *key, *value;};
struct entry table[TABLESIZE];
static int numentries=0;

void addToTable(char *key, char *value) {
  table[numentries].key=key;
  table[numentries].value=value;
  numentries++;
}

/* Find all pairs of entries in the table that sort to the same thing. */
void printAnagrams() {
  int i,j;
  for(i=0; i<numentries; i++)
    for(j=0; j<i; j++)
      if (table[i].key == table[j].key)
        printf("%s is an anagram of %s\n", table[i].value, table[j].value);
}

int main (int argc, char **argv) {
  char buf[100];
  while (gets(buf)) {
    addToTable(sort(buf), buf);
  }
  printAnagrams();
  /* don't worry about calls to free() for this exam problem */
  return 0;
}
```

Name: Student ID:

3. Assembly Language (15 points)

The following assembly code was automatically generated by compiling a very short C function that takes a single parameter of type "unsigned char *" and has a return type void.

```
      .file "question3.c"
      .text
.globl question3
      .type question3, @function
wipe:
      pushl %ebp
      movl  %esp, %ebp
      pushl %ebx
      subl  $4, %esp
      movl  8(%ebp), %ebx
      cmpb  $0, (%ebx)
      je    .L7
.L5:
      movzbl(%ebx), %edx
      movb  %dl, %al
      subb  $65, %al
      cmpb  $25, %al
      ja    .L4
      movzbl%dl, %eax
      movl  %eax, (%esp)
      call  putchar
.L4:
      incl  %ebx
      cmpb  $0, (%ebx)
      jne   .L5
.L7:
      addl  $4, %esp
      popl  %ebx
      popl  %ebp
      ret
```

Here are some hints to help understand this code:

1. "dl" and "al" are just the low-order byte names of the registers.
2. "movzbl" transfers a byte into a long register, and sets the high-order bytes to zero.
3. "ja" is the unsigned equivalent of "jg", and means "jump if above".
4. 65 is ASCII for uppercase A.

Name: Student ID:

(a) The two instructions before L5 are similar to the two before L7. In simple words, explain what they are doing? (i.e., give a brief, “high-level” description of their function). (5 points)

(b) Describe the conditions necessary for the “ja” in L5 to fail. Remember that “ja” is an unsigned comparison. (5 points)

(c) In simple words, give a high-level summary of what this code does. (5 points)

Name: Student ID:

4. Program with `fork()`. (20 points)

(a) This question concerns the following program named `ee209.c`. How many times does the program print "ee209"? Explain your answer. (6 points)

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

void ee209(void) {
  pid_t pid;

  fflush(stdout);
  if (!(pid = fork())) {
    fflush(stdout);
    fork();
    printf("ee209\n");
  }
}

int main(void) {
  ee209();
  printf("ee209\n");
  return 0;
}
```

(b) What is the role of "`fflush(stdout)`" in the above code? (2 points)

Name: Student ID:

(b) In this problem, we implement the Unix command:

```
ls | wc –l
```

This command counts the number of contents at the current directory. Fill the following 6 boxes. (12 points; 2 points for each box)

```
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char** argv) {
  int des_p[2];

  // create pipe with file descriptor “des_p”
  (i)

  if(fork() == 0) {       //first fork

    // closing stdout
    // replacing stdout with pipe write
    // closing pipe read
    (ii)

    char* const prog1[] = {"ls", 0};

    // execute command “ls”
    (iii)

    exit(1);
  }

  if(fork() == 0) {       //creating 2nd child

    //closing stdin
    //replacing stdin with pipe read
    //closing pipe write
    (iv)
```

Name: Student ID:

```
    char* const prog2[] = {"wc", "-l", 0};

    // execute command “wc -l”
    (v)
    exit(1);
  }

  // clean up pipe
  (vi)

  wait(0);
  wait(0);

  return 0;
}
```

Name: Student ID:

5. Signal. (20 points)

Write a program that enthusiastically prints the following output:

```
10… 9… 8… 7… 6… 5… 4… 3… 2… 1… Happy New Year!
10… 9… 8… 7… 6… 5… 4… 3… 2… 1… Happy New Year!
10… 9… 8… 7… 6… 5… 4… 3… 2… 1… Happy New Year!
…
```

over and over again. After printing "10… ", the program should print each element (a number with three dots, or the phrase "Happy new year!") ***one second*** (in wall-clock time) after printing the previous item, and then immediately print "10…" and so on.

The user should be able to stop the program only if Ctrl-C is typed ***twice*** within the same countdown, i.e., twice between the print of "10…" and "Happy New Year!" on the same line. Please write comments make it easier to achieve partial credit. To implement the one-second countdowns, you can use the *sleep()* and *alarm()*.

Start with following library.

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <assert.h>
#include <unistd.h>
```

Name: Student ID:

Don’t give up!