CSI 402 – Systems Programming – Spring 2014
Midterm Examination – March 11, 2014

Note: This examination has **four** questions for a total of 100 points. **Answer all questions. Write all the answers on your blue book.**

Question I (44 points total)

(a) Does MIPS represent an example of a load-store architecture? Justify your answer. (2 points)

(b) The Sun SPARC architecture allows register windows assigned to different procedures to overlap. Mention one advantage of this scheme. (2 points)

(c) Starting from an empty binary search tree, suppose we insert six symbols into the tree in the following order: `GOOD`, `LIKE`, `BUFFER`, `BAD`, `CROWD` and `VALUE`. Show the resulting binary search tree and indicate its height. *You need to show only the final tree and its height*; there is no need to show the intermediate trees. (6 points)

(d) Suppose memory addresses 21, 33 and 2700 (all decimal) of a SIC/XE machine contain the values 2400, 2700 and 33 (all decimal) respectively. The `A` register (i.e., the accumulator) contains the value −17 (decimal) when the machine executes the 3-byte instruction whose hexadecimal representation is `320021`. Show the contents of the memory locations 21, 33 and 2700 after the execution of the instruction. Your answers must be in *decimal.* Show work. (10 points)

Note: The 6-bit opcode `001100` corresponds to the `STA` (Store Accumulator) instruction.

(e) Consider the following instruction for SIC/XE:

```
JEQ   LOOP
```

The LC-value for the above instruction is 250 (decimal) and the LC-value for the symbol `LOOP` is 234 (decimal). Assuming that the instruction is being assembled using PC-relative mode, show the assembled form of the instruction in *hexadecimal* form. The 6-bit opcode for `JEQ` is `110000`. Show work. (12 points)

(f) Consider the following module written in SIC/XE Assembly language.

```
Q1          START    0
FIRST      +LDA      VAL
            STA      BLOCK
            LDX      #0
LOOP        ADD      BUF
            TIX      #5
            JLT      LOOP
           +STA      VAL
BLOCK       RESW     1
BUF         WORD     5
VAL         WORD     13
            END      FIRST
```

Show the symbol table produced by a 2-Pass assembler for the above module. The LC-values shown in the Symbol Table must be in *decimal.* (12 points)

Question II (21 points total)

(a) Assume that the file "`infile.txt`" contains only the following line:

```
abcdefghij\n
```

where `'\n'` is the (single) newline character. Indicate the output produced by the following program, assuming that the call to `fopen` does not fail. **No explanation is needed.** (9 points)

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
  FILE *ifile;   char  c;   long  pos;
  if ((ifile = fopen("infile.txt", "r")) == NULL) {
     fprintf(stderr, "Cannot open infile.txt.\n"); exit(1);
  }
  fseek(ifile, -5L, SEEK_END);  pos = ftell(ifile);  c = fgetc(ifile);
  printf("%ld  %c\n", pos, c);
  fseek(ifile, -4L, SEEK_CUR);  pos = ftell(ifile);  c = fgetc(ifile);
  printf("%ld  %c\n", pos, c);
  rewind(ifile);  pos = ftell(ifile);  c = fgetc(ifile);
  printf("%ld  %c\n", pos, c);  return 0;
} /* End of main. */
```

(b) A C program has been split into two source files called `main.c` and `funct.c`. The contents of these two files are shown below.

File: `main.c`

```
#include <stdio.h>
int p, q;
void mystery(int);
int main(void) {
  int r = 2;  p = -7;  q = 5;
  while (r > 0) {
     printf("%d  %d  %d\n", p, q, r);  mystery(r);
     printf("%d  %d  %d\n", p, q, r);  r--;
  }
  return 0;
}
```

File: `funct.c`

```
extern int q;
void mystery(int x) {
  int p, r;
  p = x++;  r = 2 * p;  q = 2 + r;
}
```

There are no syntax errors in either of the above C files. The executable version (`a.out`) of the program is created using the following Unix command:

```
        gcc  main.c  funct.c
```

Indicate the output produced when `a.out` is executed. **No explanation is needed.** (12 points)

Question III (15 points)

This problem involves a binary search tree where each node stores a symbol (a string of length *at most* 15) along with pointers to its left and right children. Thus, the `struct` definition for each node of the tree is as follows:

```
   #define  SIZE  15
   struct  tree_node {
       char  symbol[SIZE+1];
       struct tree_node *left_child, *right_child;
   };
```

Write a function with the following header:

```
        int  count (struct tree_node *rp, int maxlen)
```

Here the parameter `rp` is a pointer to the root node of a binary search tree. Your function must return the number of *leaf nodes* of the tree where the *length of the string stored is at most the value given by the parameter* `maxlen`.

You need to show only the C code for the above function. You may use magic numbers and there is no need to include comments in your code.

Question IV (20 points)

Assume that the following constant and type definitions (for storing information about employees in a company) are available.

```
#define   NAME_MAX   30
#define   TITLE_MAX  20
struct   employee {
  char  name[NAME_MAX];        int  id_number;
  char  job_title[TITLE_MAX];  int  age;  float salary;
};
```

Write a function `create_manager_file` with the following header:

```
      void  create_manager_file (char *infile,  char *outfile)
```

Here, `infile` is the name of a *binary* (unformatted) input file which contains zero or more records of type `struct employee`. The parameter `outfile` is the name of a *text* file to be created by the function. Your function must go through the records of the input file. For each record in the input file where the value of the `job_title` field is the string `"Manager"`, your program must write to the output file the values of the following fields: `name`, `id_number` and `salary`. Each line of the output file must contain the required information for *exactly one* employee.

Your answer needs to contain only the C code for the above function. No error checks are necessary. You may use magic numbers and there is no need to include comments in your code.