Extra Review Questions

1. Basic C

   a) What does the following code fragment print?

```
for (i = 0; i < 100; i++)
    printf("%d\n", i += i);
```

   b) What is the difference between these two code fragments?

```
(A)     for (i = 2; i < 9; i += 2)
            func(i);

(B)     i = 3;
        while (i <= 9)
        {
            i++;
            func(i-2);
            i++;
        }
```

   c) What is the value of j after this code is executed?

```
for (i = 1, j = 2; i < 6; i++) j += i;
```

2. TOY

   a) What does the following code fragment do?

```
1112
2212
2112
```

   b) Give the hex value left in R1 by the following TOY program:

```
10: 9110
11: 9211
12: 2121
13: 9312
14: 1113
15: 0000
```

   c) All but one of these TOY instructions, if executed at location 10, have the same ultimate effect on R1, no matter what it contains prior to executing the instruction. Which is the exception?

```
A. 2111
B. D111
C. 7110
D. E911
E. B100
```

3. functions

   a) What value is printed by the following C program?

```
#include <stdio.h>
int f(int x)
    {     return 3*x + 1;  }
void main()
    {
        printf("%d\n", f(f(2)));
    }
```

b) What value is printed by the following C program?

```
#include <stdio.h>
int f(int x)
    { return 3*x + 1; }
int g(int x)
    { return f(2*x) - f(x); }
void main()
    { printf("%d\n", f(g(2))); }
```

c) What value is printed by the following C program?

```
#include <stdio.h>
int f(int x)
  { return x * 2 - 1; }
void main()
  { printf("%d\n", f(f(f(f(1))))); }
```

d) What is the effect of this recursive function?

```
int strange (int x)
{
    if ((x % 2) != 0) return x;
    return 2 * strange(x/2);
}
```

e) Consider the following pair of mutually recursive functions. What is the value of f (3) ?

```
int f (int x)
    { if (x > 0) return g(x) + f(x-1);  return 1; }
int g (int x)
    { if (x > 0) return g(x-1) + f(x-1) - 1;   return 1; }
```

4. arrays

a) What values are printed by the following C program?

```
#include <stdio.h>
void main()
{
    int i, a[10];
    a[0] = 1; a[1] = 1;
    for (i = 2; i < 10; i++)
    {
           a[i] = a[i-1] + a[i-2];
           printf("%d ", a[i]*a[i-2] - a[i-1]*a[i-1]);
    }
    printf("\n");
}
```

b)
What values are printed by the following C program?

```
#include <stdio.h>
void main()
{
    int i, j, a[8]; int k;
    for (i = 0; i < 8; i++) a[i] = 1;
    for (i = 1; i < 7; i++)
           for (j = i; j > 0; j--)
           {  a[j] = a[j] + a[j-1]; }
    for (i = 0; i < 8; i++)
           printf("%d ", a[i]);
    printf("\n");
}
```

c) Write a C program to find the average of the element of an array `A[0..n-1]` and to print the index of the element that is closest to average (a tie is broken in favor of the element appearing first). You may assume that $n \geq 1$, so you don't need to check for an empty array.

d) What is the running time of the entire program?

e) Suppose that a huge array contains integers in the range 0 to 50. Give C code that prints the value(s) that occurs most often in the array. Assume that the array is named **a** and its size is a defined constant **N**.

5. structs

a) Define a struct suitable for representing a playing card.

b) Using your definition of a playing card struct, write a function called 'less' that, given two cards, returns '1' if the first card is less than the second in value and '0' otherwise. Aces are ranked high, and spades > hearts > diamonds > clubs.

c) Write a function that, given an array c[] of cards and an int n, the number of cards in the array, will return the highest ranked card. You may assume that c[] has at least 1 card.

d) Using the same assumptions as in part c, write a function to sort a 'hand' of cards.

6. linked lists: for these questions, assume the following definition of nodes on the lists:

```
typedef struct node *link;
struct node {
       int key;
       link next;
};
```

a) Assume that **list** is a pointer of type **link** that points to a list with at least one node on it. Write a code fragment to add a new node with key value 999 just after the first node of the list.

b) Write a function that takes a link to a list as its argument and returns a link to the first node on the list with key value 0 (NULL if there is no such node.)

c) Write a recursive function that will remove all list elements with key value equal to 0. You do not need to free the nodes.

d) Write a recursive function that returns the sum of the keys on a linked list.

e) Write a recursive function that deletes every other element of a linked list. Example:

f) Assume that the nodes on a linked list are sorted in ascending order. Write a recursive function that returns **1** if a given value is in the *sorted* linked list and **0** otherwise. (Assume the list is sorted in ascending order.)

g) Write a recursive function that creates a new node for a given value and inserts that node into a *sorted* linked list. (Hint: here's a prototype for this function.)

```
link insert (int value, link list);
```

7. circular linked lists

a) Write a function that returns the number of nodes on a circular list, given a pointer to one of the nodes on the list.

b) Write a function that determines the number of nodes that are between the nodes referenced by two given pointers x and t to nodes on a circular list.

c) Write a code fragment that, given pointers x and t to two disjoint circular lists, inserts the list pointed to by t into the list pointed to by x, right after the first node of the list pointed to by x. You may assume that there is at least one node on each of the lists.

d) Given pointers x and t to nodes on a circular list, write a code fragment that moves the node following t to the position following the node following x on the list. You may assume that the node pointed to by x, the node pointed to by t, and the node following t are all distinct nodes.

8. binary tree exercises: assume the following definition of a node.

```
typedef struct node * link;
struct node {
    int key;
    link left;
    link right;
};
```

a) Write a function that counts the leaves of a binary tree.

b) Write a function that deletes all leaves of a binary tree that have their key equal to a given value.

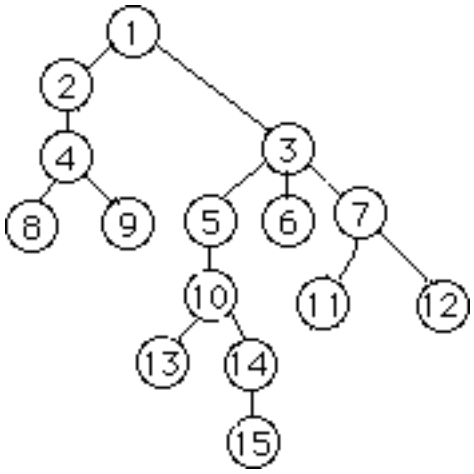
c) Write a function that returns the sum of all the keys of a binary tree.

9. For these exercises, suppose a tree (*not necessarily binary*) is built with nodes defined as:

```
typedef struct node *link;
struct node {
        int key;
        link leftmostChild;
        link rightSibling;
};
```

(See Sedgewick Property 5.4)

a) Give the binary tree representation of the tree pictured here:



b) Given the following implementation of a preorder traversal, list the nodes of the tree above in preorder.

```
void preorder(link n)
{
        link c;
        printf("%d\n", n->key);
        c = n->leftmostChild;
        while (c != NULL) {
                preorder(c);
                c = c->rightSibling;
        }
}
```

c) How could you change the above function to print the nodes in postorder? List the nodes of the tree above in postorder.

10.

a) Evaluate the prefix expression: * + * + * 6 5 4 3 2 1

b) Convert the expression a b + c * d e - / f + from postfix to (a) infix and (b) prefix.

c) Evaluate this prefix expression: / * * 2 + 2 3 6 5 and write the equivalent postfix expression.

11. What does the following function do?

```
void squeeze (char s[], int c)
{
    int i,j;

    for (i = j = 0; s[i] != '\0'; i++)
           if (s[i] != c)
                  s[j++] = s[i];
    s[j] = '\0';
}
```

12. FSAs/REs

a) Draw a deterministic FSA that accepts all strings of 0's and 1's which contain the substring 1001.

b) Give the regular expression which describes the language in a.

c) Draw a deterministic FSA that accepts all strings of 0's and 1's which DO NOT contain the substring 1001. (Hint: your machine should be very similar to the one in a).

d) Draw a deterministic FSA which accepts all strings of 0's and 1's which contain an odd number of 0's.

e) Give the regular expression describing the language in d.

f) Draw a deterministic FSA which accepts all strings of 0's and 1's which contain an odd number of 0's and an odd number of 1's. (Hint: it should have four states).

13. More regular expressions!

a) The regular expression (a + ab)(c + bc) describes the language that consists of the three strings **ac**, **abc**, and **abbc**. Write two other regular expressions that define the same language.

b) Write a RE that defines the language of all strings of a's and b's such that all runs of a's are of even length. That is, strings such as bbbaabaaaa, aaaabb, and the empty string are in the language; abbabaa and aaa are not.

c) Write a RE that defines the language of strings that represent numbers of type **float** in C.

d) Write a RE that defines the language of strings of 0's and 1's having an even number of 1's.

e) Describe the language defined by the RE: (a + b)*

f) Describe the language defined by the RE: (a*ba*b)*a*

g) Which of the following REs does *not* describe the same language as all of the others

```
A.   1(0+1)*
B.   1(0+1)*(0+1)*
C.   1 + 1(00 + 01 + 10 + 11)*
D.   1(0*1*)*
E.   1(0* + 1*)*
F.   1 + 1(0+1)*0 + 1(0+1)*1
```

14. Convert the following type 3 grammars to FSAs

a)

```
A -> z0
z -> A1
z -> e

A and z are the non-terminals.
A is the start non-terminal.
e is the empty string.
```

b)

```
A -> B01
B -> C0
C -> A1
C -> 0

A, B, and C are the non-terminals.
A is the start non-terminal.
```

15. Context Free Grammars

a) What language does the following grammar describe:

```
S -> A
A -> 1A1 | 0A0 | 2
```

b) . Draw a deterministic PDA which accepts the language in a.

c) Was it Eliot's toilet I saw?

d) What language does the following grammar describe:

```
S -> ABA
A -> 1A | 0A | e
B -> 1001
```

(where e is the empty string).

e) Is the language in d regular? Why or why not?

16. In C, an 'identifier' is a string of letters and digits, beginning with a letter (or underscore). Give a grammar to define the set of all C identifiers. (You may ignore the fact that some reserved words such as 'for', 'while', 'struct' etc. are not allowed.)

17. For the following exercises, if your answer is 'no', give an explanation. if your answer is 'yes', show how to create the FSA, PDA, or C program.

a) Can you create a FSA that determines whether the parentheses in an expression are balanced?

b) Can you create a PDA that determines whether parentheses in an expression are balanced?

c) Can you write a C program that determines whether parentheses in an input expression are balanced?

18. Suppose that a being from another planet landed at the United Nations and handed the citizens of earth a polynomial-time deterministic algorithm for the SATISFIABILITY problem. How would you use it to solve an instance of the WIZBAN problem, which is in NP, efficiently?

19.Strings:

a) What is wrong with this program?

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

int main(void)
{
        char str[10];

        strcpy(str, "this is a test");
        printf(str);
}
```

b) Write a program that reads any number of strings from the command line and prints them backwards & in reverse order.
for example: **a.out hello how are you**
program prints: **uoy era woh olleh**

c) Write a function that takes two strings (s1 & s2) as its inputs and deletes each character in s1 that matches any character in the string s2.

d) Write a function that takes two strings (s1 & s2) as its inputs and returns the first location in the string s1 where any character from the string s2 occurs, or -1 if s1 contains no characters from s2.

20. Consider the following function:

```
void bs(int a[], int asize)
{

       int i, j, temp;

       for(i = asize-1; i > 0; i--)
       {
              for(j = 0; j < i; j++)
              {
                     if (a[j] > a[j+1])
                     {
                            temp = a[j];
                            a[j] = a[j+1];
                            a[j+1] = temp;
                     }
              }
       }
}
```

a) What does this function do to the array a?

b) How many times is the if statement executed?
(as a function of asize)

c) What is this in big O notation?
(again as a function of asize)

d) Write a swap function that swaps two places in an integer array.
(it should do the same thing as the line after the if)

Hint: The prototype for swap is: void swap(int *, int *);

21. Consider the factorial program fragment below, and let input size be the value of *n* that is read. Counting one time unit for each assignment, read, and write statement, and one unit each time the condition of the while-statement is tested, compute the running time of the program.

```
scanf("%d", &n);
i = 2;
fact = 1;
while (i <= n) {
  fact = fact*i;
  i++;
}
printf("%d\n", fact);
```

22. Simulate the Selection sort algorithm (Sedgewick section 6.2) on an array containing the elements

a) 6, 8, 14, 17, 23

b) 17, 23, 14, 6, 8

c) 23, 17, 14, 8, 6

· How many comparisons and swaps of elements are made in each case?

23. Insert sort

a) Write a function which performs insert sort on an array of integers (you may use swap).

b) For a list of numbers of size n:
What is the best case number of comparisons that insert sort to sort them? Give an example of a best case order.

What is the worst case number of comparisons that insert sort takes to sort them? Give an example of a worst case order.

c) Show the results of each pass (outer loop) when the following list is insert-sorted:

```
6 7 5 3 0 9
```

24. Quick sort

a) For a list of numbers of size n:
What is the best case number of comparisons that quick sort takes to sort them? Give an example of a best case order.
What is the worst case number of comparisons that quick sort takes to sort them? Give an example of a worst case order.

b) Why doesn't this mean that quick sort is worse than insert sort in general?

25. Binary Search Trees

a) Give the binary tree that results when the following letters are inserted into a binary search tree:

```
ILOVETELLYTUBBIES
```

b) Give the inorder traversal of the tree you created.

c) Give an ordering of the above letters which would result in a worst case binary tree (one with maximal depth).

d) How many comparisons are necesary to quicksort the sequence of letters in your answer to part b?

e) Given the following type definition:

```
typedef struct node *link;
struct node {int value; link left; link right;};
```

write a recursive function that returns the sum of the values in a tree.

f) Write a recursive function that determines the depth of a tree

26. Construction of a parse tree (Sedgewick Program 5.20)

```
char *expression;
int cur_position;

typedef struct node * link;

struct node {
        char token;
        link l;
        link r;
};

link parse()
{
        // get next token
        char t = expression[cur_position];
        cur_position++;

        // create new node for token
        link x = malloc(sizeof *x);
        x->token = t;
        x->l = NULL;
        x->r = NULL;

        if ((t == '+') || (t == '*') || (t == '-') || (t == '/'))
        {
                x->l = parse();
                x->r = parse();
        }
        return x;
        }
```

(assume that all expressions consist only of single digit values on the operators: +,-,*,/)

a) Trace through code for the expression: "-*+345/82" and show the resulting tree.

b) Write a recursive function to traverse the tree and print the expression in inorder notation. For example, the above expression should be printed as: "(((3+4)*5)-(8/2))"

c) Write a recursive function to calculate the value of the expression. You may assume that the tree is not empty.

27. Build abstract syntax trees for each of the following expressions. Then traverse these syntax trees in postorder to produce the corresponding reverse polish notation expressions. Third, generate TOY code for the expressions by hand-simulating the compiler code (given in the compiler lecture, available from the 'lectures' page on the course website.)

a) (a + 1) * ((a - b) + 4)

b) (9 * 8) + (7 * 6) + 5

c) 1 + (2 * 3) - (4 * (5 + 6))

28. Create a lexical analyzer function `int getToken(FILE *in, char s[])` which reads the file in (which has already been opened etc.) and returns the next token. Tokens are defined as follows:

- identifiers: a sequence of characters that begins with a letter followed by an arbitrary number of letters or digits.
- integer constants: a sequence of digits
- single-character operators: '(', ')', '+', '-', '*' and '/'

The return value should indicate the type of token recognized (0 for identifiers, 1 for integer constants, 2 for operators, and -1 for an unrecognized token) and the `string s[]` should contain the token recognized.

The lexical analyzer should eat up blank spaces (' ' and '\n').

Hint: in order to determine the end of a token you need to read one character past the end of a token. This character may be part of another token, so it should be put back in the stream. You can use the C function ungetc(char, FILE *) for this.

29. We decided to add multiprogramming support to TOY. To do this, 1) we added timer interrupts to TOY, 2) we wrote a small TOY Operating System (TOS) that takes over control when a timer interrupt occurs.

a) What does the timer interrupt handler do?

b) Why is this not sufficient for running multiple instances of the same TOY program simultaneously?

c) Name the operating system mechanism that is necessary to fix this problem.

30. Process Scheduling: show the CPU utilization by the following processes assuming

a) first-come-first-serve

b) shortest job first

c) round-robin with quantum = 10 time units

d) round-robin with quantum = 8 time units

`P1: 16, P2: 3, P3: 7, P4: 80, P5: 11`

- also show the average wait time for each scheduling strategy.
- discuss the pros & cons of each of the above strategies both for this particular sequence of processes and in general.

31. Page replacement

In systems with virtual memory, there is usually a relatively small main memory, which holds only some memory pages, and a large "swap space" on disk, which holds all of them. When some program references a page which is not in main memory, this page has to be brought in from the disk (this is known as a "page fault"). Unfortunately, another page will then have to be evicted from main memory, to make room for the new one. It is the job of the operating system to decide which page to evict. Two well - known policies used for this puprose are the following:

FIFO : evict the page which has stayed the longest in main memory.
LRU: evict the page that has not been referenced for the longest time.

Imagine a system with 5 virtual memory pages, and room for just 3 pages in main memory. Consider the following string of memory references:

0 1 2 3 0 1 4 0 1 2 3 4

a) Simulate the FIFO and LRU algorithms for the above string of references.
   - How many page faults do we get in each case?
   - Which pages are in main memory after this string of references in each case?

b) Do the same with a main memory holding 4 pages. (Notice that the FIFO algorithm gets more page faults in this case! This counterintuitive phenomenon is known as Belady's anomaly.)

c) Given the above results, which of the two policies would you implement in your own operating system?

32. Java questions

a) What does "null" represent in Java?.

b) What are "constructors," and how are they used?

c) What is the relationship between "classes" and "objects"? What is the difference between them?

d) Define the term "instance variable."

e) Explain what the term "static" means when used as a modifier on a variable or method declaration in a class.

f) To compute the square root of x in a Java program, you would use Math.sqrt(x). How do you interpret the name, "Math.sqrt"? What is "Math"? What is "sqrt"?

g) Explain what is meant by the terms "subclass" and "superclass."

h) What is the difference between applications and applets?

i) What are bytecodes?

j) Java uses "garbage collection" for memory management. Explain what is meant here by garbage collection. What is the alternative to garbage collection?

33. Writing java code

a) For this problem, you should write a very simple but complete class. The class represents a counter that counts 0, 1, 2, 3, 4,.... The name of the class should be *Counter*. It has one private instance variable representing the value of the counter. It has two instance methods: *increment()* adds one to the counter value, and *getValue()* returns the current counter value. Write a complete definition for the class, *Counter*.

b) The following program segment is meant to simulate tossing a coin 100 times. It should use two *Counter* objects, *headCount* and *tailCount*, to count the number of heads and the number of tails. (The Counter class is defined in part c). Fill in the blanks so that it will do so.

```
Counter headCount = new Counter();
Counter tailCount = new Counter();

for (int flip = 0; flip < 100; flip++) {
    if (Math.random() < 0.5)
          _____________________ ;   // count a "head"
    else
          _____________________ ;   // count a "tail"

System.out.println("There were " + _____________________ + " heads.");
System.out.println("There were " + _____________________ + " tails.");
```

c) Suppose that the class Employee is defined as follows:

```
class Employee {
        String name;
        double hourlyWage;
}
```

Consider an array, *emp*:

```
        Employee emp = new Employee[1000];
```

Suppose this array has **already** been filled with data about 1000 employees. Write a code segment that will count the number of employees who earn more than 25.00 per hour.

d) Suppose that you want a very simple class to represent the money in a bank account. It needs three methods, one to deposit a given amount into the account, one to withdraw a given amount, and one to check how much money is in the account. It also needs a constructor that creates an account containing a specified initial amount of money. The class should have one instance variable, for storing the amount of money in the account. Write a complete Java class satisfying this requirement.

e) What is output from the following program?

```
class AnotherObject {
    int i;
    String s;
    double d;

    AnotherObject(int i, String s, double d) {
        this.i = i;
        this.s = s;
        this.d = d;
    }

    void display() {
        System.out.println("i = " + i + "; s = " + s + "; d = " + d);
    }
}

class Test {
    public static void main(String args[]) {

        AnotherObject ao1 = new AnotherObject(5, "Hello", 3.4E100);
        AnotherObject ao2 = ao1;

        ao1.i = 3;
        ao1.s = "New string";
        ao1.d = 6.02E23;

        ao1.display();
        ao2.display();
    }
}
```

34. Java inheritance: Consider the following Java code. What is the output produced by the "main" program?

```
public class A {
    int x,y;
    public A() {
       x = 1;
       y = 2;
    }
    public int foo(int z) {
       x = z;
       return x*y;
    }
    public int bar() {
       return x;
    }
}

public class B extends A {
    int x;
    public int foo(int z) {
       x = z;
       return x+y;
    }
    public int baz() {
       return x;
    }
}

class Test {
    public static void main(String[] args) {
       B b = new B();
       System.out.println(b.foo(3));
       System.out.println(b.foo(4));
       System.out.println(b.bar());
       System.out.println(b.baz());
    }
       }
```