

# ECE 199 Final Exam Spring 2003

Monday, May 12<sup>th</sup>, 2003

Name:

- **Be sure your exam booklet has 8 pages.**
- **Write your name at the top of each page.**
- **This is a closed book exam.**
- **You are allowed three 8.5 x 11 sheets of notes.**
- **Absolutely no interaction between students is allowed.**
- **Show all of your work.**
- **Don't panic, and good luck!**
- **Challenge questions are marked \*\*\***

Problem 1	20 points	_____
Problem 2	20 points	_____
Problem 3	20 points	_____
Problem 4	20 points	_____
Problem 5	20 points	_____
Total	100 points	_____

Name: \_\_\_\_\_

**Problem 1 (20 points): Short Answer**

Answer the following questions with a **very brief and precise** statement. A long statement is a good indication that your response will be marked incorrect. As a general guideline, each correct response should require no more than 20 words.

**Part A** (6 points): Recall from MP4 we made use of a very simple dictionary-based compression scheme where the 52 most frequent words were encoded using 2 character keywords. Using this scheme as a starting point, describe an optimization to improve the compression ratio (i.e., make the compressed file smaller).

**Part B** (6 points): Consider a sequence of  $n$  sorted data items. Would it be better to store these items in an array or in a linked list if the most frequent operation we will perform on these items is the search operation? Justify your answer.

**Part C** (8 points): Rewrite the following code so that it does not use an **if-else** statement. Any other legal C constructs (**if**, **for**, **while**, etc...) can be used instead. All variables are of integer type.

```
if (b == 0) {
    x = x + 1;
}
else if (a == 0) {
    y = y + 1;
}
```

Name: \_\_\_\_\_

**Problem 2** (20 points): Debugging.

**Part A** (8 points): The code snippet below contains a flaw. Identify it.

```
;R0 will contain the sum upon completion of SUM_ARRAY
;R1 contains address of start of array
;R2 contains number of elements in the array
SUM_ARRAY ST R1, SAVE_R1      ;Save Registers
          ST R2, SAVE_R2
          ST R3, SAVE_R3
          AND R0, R0, #0
NEXT      ADD R2, R2, #-1      ;More numbers?
          BRnz DONE
          LDR R3, R1, #0
          ADD R0, R0, R3      ;Add this new number
          ADD R1, R1, #1
          BRnzp NEXT
DONE      LD R1, SAVE_R1      ;Restore Registers
          LD R2, SAVE_R2
          LD R3, SAVE_R3
          RET
```

**Part B** (8 points) When the following code is run on an LC-2 system, it never terminates. Explain why.

```
#include <stdio.h>

int main()
{
    int x[99];
    int y;
    int sum = 0;

    for( y = 0; y <= 99; y++ ) {
        x[y] = y / 2;
        sum = sum + x[y];
    }

    printf( "%d\n", sum);
    return 0;
}
```

Name: \_\_\_\_\_

\*\*\*Part C (4 points) A hardware test engineer noticed strange behavior on a prototype LC-2 system while running the below code. Using the collected data below, deduce and describe the hardware problem that could cause such behavior. Note: It is just a simple **single** error.

```
.ORIG x3000
LD R0, x50
ADD R0, R0, #0
ST R0, x50
.END
```

**Before the LD executes**  
R0: x0000

Memory	
x3050	x5000
x3051	x6000

**After the ST executes**  
R0: x6001

Memory	
x3050	x5000
x3051	x6001

Name: \_\_\_\_\_

**Problem 3 (20 points):** From LC-2 to C

**Part A** (10 points). In Column A below is a list of LC-2 constructs. Some are single instructions, some are short code sequences. Column B contains C programming constructs. Draw arrows from each LC-2 construct to the C constructs that directly use that construct. Items from Column A might have more than one arrow leading out of it and items from Column B might have no arrows leading to them.

Column A	Column B
JSR	while statement
LDR R3, R6, #5 LDR R0, R3, 0	ptr->next
BRnz LABEL_A	if statement
LDR R3, R6, #5 LDR R0, R3, 4	recursion
BRnzp LABEL_B	functions
	if-else statement
	y = *x;

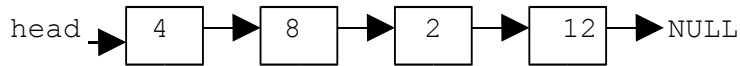
**Part B** (10 points). Provide the value of the following expressions given the state of memory as shown. Variable **x** is declared as an **int \*** and is allocated at memory location **x4002**.

(i)	x	x4002	x4003
(ii)	*x	x4003	x400A
		x4004	x4006
		x4005	x400F
(iii)	&x	x4006	x4007

Name: \_\_\_\_\_

**Problem 4 (20 Points):** Linked Lists

**Part A** (15 points): Given a pointer to the head node of a linked list, print out the elements of that list in reverse order. An example is shown below. The `list_node` structure is defined below. Hint: it is possible to do this with fewer than 8 additional lines of code.



**example output:**

12  
2  
8  
4

```
struct list_node {  
    int value;  
    struct list_node *next;  
};  
typedef struct list_node List;  
  
void PrintReverse(List *head)  
{
```

```
}
```

Name: \_\_\_\_\_

**Part B** (5 points): **ListFunction** takes two linked lists (that is, two head pointers to something of type **List**), does something to them, and returns a pointer to something of type **List**. Describe what it does in 2 or 3 sentences. Assume that the input lists have already been sorted in increasing order.

```
struct list_node {
    int value;
    struct list_node *next;
};
typedef struct list_node List;

List *ListFunction(List *headA, List *headB)
{
    List *result = NULL;

    if (headA == NULL)
        return headB;
    else if (headB == NULL)
        return headA;
    else {
        if (headA->value <= headB->value) {
            result = headA;
            result->next = ListFunction(headA->next, headB);
        }
        else {
            result = headB;
            result->next = ListFunction(headA, headB->next);
        }

        return result;
    }
}
```

Name: \_\_\_\_\_

### Problem 5 (20 points): Arrays

The following function performs a string manipulation that might be very useful for a function like `printf`. This function inserts `stringB` somewhere within `stringA`, as indicated by the appearance of a `'%'` character within `stringA`. For example, if `stringA` equals "Today is the % day of May", and `stringB` is "twelfth", then the function will modify `stringA` to contain: "Today is the twelfth day of May". Assume the array of `stringA` is large enough to hold the additional characters from `stringB`. Also, neither string will be empty, but `stringA` might not contain the `'%'` character. Provide the missing pieces of code in the underlined, blank lines below.

```
void insertString(char stringA[], char stringB[])
{
    int insertPoint, lengthA, lengthB, jj;

    insertPoint = 0;
    while (stringA[insertPoint] != '%' && stringA[insertPoint] != '\0')
        insertPoint++;

    lengthA = 0;
    while (stringA[lengthA] != '\0')
        lengthA++;

    lengthB = 0;
    while (stringB[lengthB] != '\0')
        lengthB++;

    if (insertPoint == _____)
        return;

    for(jj = lengthA; jj >= insertPoint; jj--)
        stringA[_____] = stringA[_____];

    for(jj = 0; jj < lengthB; jj++)
        stringA[_____] = stringB[_____];
}
```