

ECE 199 Final Exam Fall 2005

Friday, December 16th, 2005

Name:

- **Be sure your exam booklet has 13 pages.**
- **Write your name at the top of each page.**
- **This is a closed book exam.**
- **You may not use a calculator.**
- **You are allowed three handwritten 8.5 x 11” sheets of notes.**
- **Absolutely no interaction between students is allowed.**
- **Show all of your work.**
- **Be sure to clearly indicate any assumptions that you make.**
- **Don’t panic, and good luck!**

“Some talk of an appeal unto some passion,
Some to men’s feelings, others to their reason:
The last of these was never much the fashion,
For reason thinks all reasoning out of season.”

– Lord Byron

\		
Problem 1	20 points	_____
Problem 2	20 points	_____
Problem 3	20 points	_____
Problem 4	20 points	_____
Problem 5	20 points	_____

Total 100 points

Problem 1 (20 points): Short Answer

Part A (3 points): You use a website called “Facebook” which maintains a list of people you claim to be friends with. Your list currently contains 357 friends (conveniently listed in alphabetical order). You want to read the profile of each of them. If you plan to read them in order, and would like a way to store which of the 357 profiles you read last, how many bits of storage do you need? Justify your answer.

Part B (2 points): If you would also like a way to store which of the 357 “friends” you have actually met from **Part A**, and which you have not, how many additional bits do you need? Justify your answer.

Part C (5 points): You are stranded on a deserted island with a box of a single type of logic gate and a broken motorboat. To escape from the island, you must design a logic controller for the motorboat. The controller requires that you implement an arbitrary Boolean logic function. Why would you prefer a box of NAND gates to a box of AND gates? Justify your answer.

Problem 1, continued

Part D (5 points): Give one advantage of using an enumeration (`enum`) rather than using integers directly to represent a set of values in C.

Part E (5 points): The `pick_display` function shown below has a bug and sometimes crashes. Describe the problem and how it can be fixed (in words, not code).

```
static unsigned char digit[10] = {
    0x77, 0x41, 0x6E, 0x6B, 0x59, 0x3B, 0x3F, 0x61, 0x7F, 0x79
};

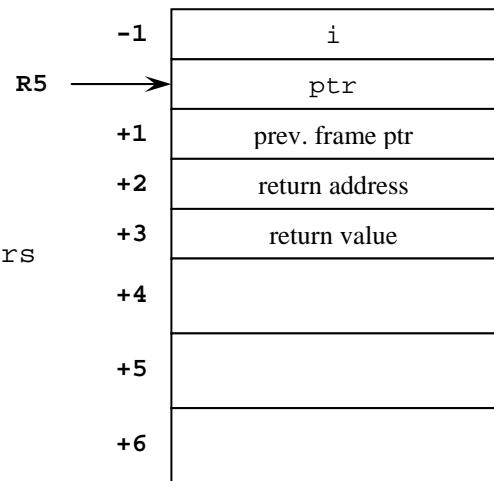
unsigned char
pick_display ()
{
    int num;

    printf ("What number should appear on the 7-segment display? ");
    if (1 != scanf ("%d", &num)) {
        num = 0;
    }
    return digit[num];
}
```

Problem 2 (20 points): Assembly and Programming Basics

This problem uses structures similar to those used in the first checkpoint of MP2. The code below parses an array of C strings consisting of letters marking open directions and exits for each location in the maze and transforms these strings into an array of bit vectors representing the same information more compactly.

```
enum {LEFT = 1, RIGHT = 2, UP = 4, DOWN = 8, EXIT = 16};
int create_vectors (int num, char* maze[], unsigned short bit_vector[])
{
    int i;
    char* ptr;
    for (i = 0; num > i; i++){
        bit_vector[i] = 0;
        for (ptr = maze[i]; '\0' != *ptr; ptr++) {
            switch (*ptr){
                case 'l': bit_vector[i] |= LEFT; break;
                case 'r': bit_vector[i] |= RIGHT; break;
                case 'u':
                    bit_vector[i] |= UP; /* Part B */
                    break;
                case 'd': bit_vector[i] |= DOWN; break;
                case 'e': bit_vector[i] |= EXIT; break;
                default: break;
            }
        }
    }
    return 1;
}
```



Part A (3 points): Complete the stack frame shown to the right (also called an activation record) for `create_vectors` by writing the parameter names into the blanks in the appropriate order.

Part B (8 points): Translate **the single line** marked by the comment (“`/* Part B */`”) into LC-3 assembly code.

Problem 2, continued

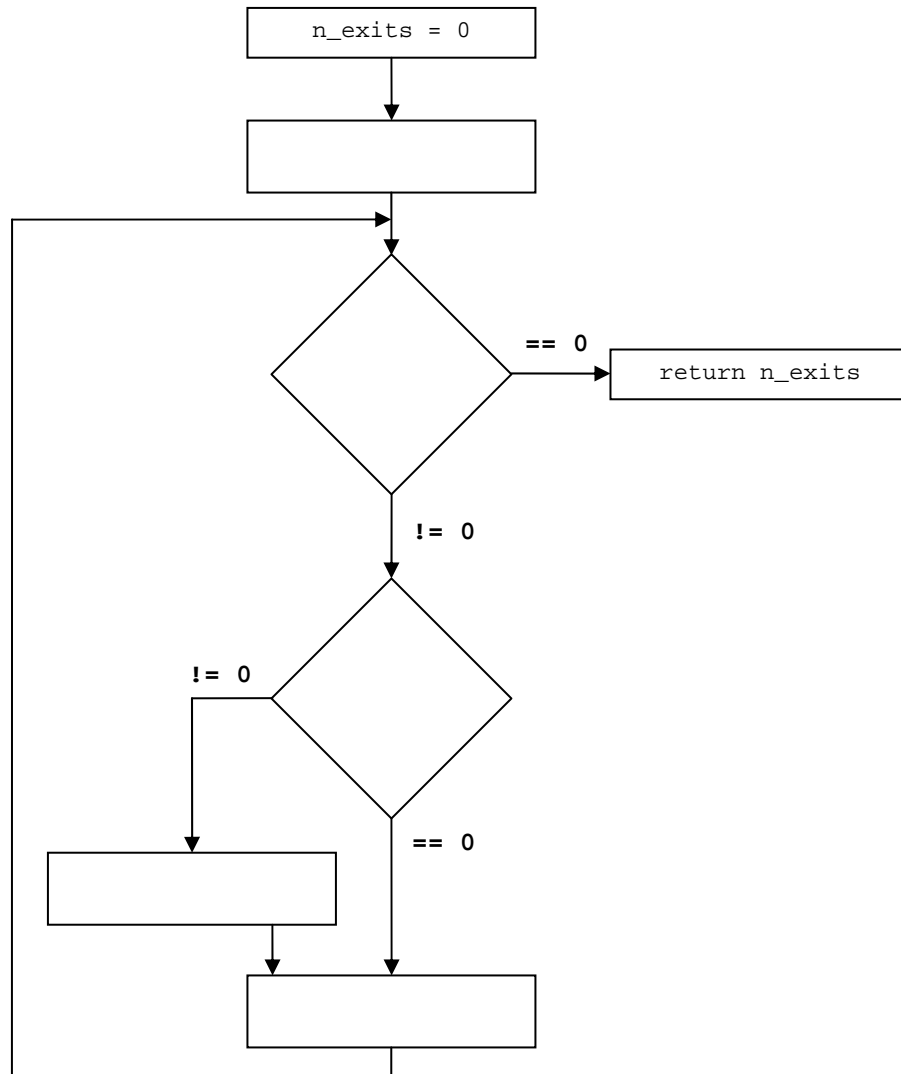
Part C (4 points): How many memory accesses are required for the LC-3 to execute your code from **Part B**? Justify your answer.

Part D (5 points): Now consider a function that counts the number of exits in a maze. Fill in the flow chart shapes below with C statements and expressions to complete the systematic decomposition of the problem.

```
enum {LEFT = 1, RIGHT = 2, UP = 4, DOWN = 8, EXIT = 16};
```

```
int
count_exits (int num, unsigned short bvec[])
{
```

```
    int n_exits;
    int i;
```



```
}
```

Problem 3 (20 points): C Structures, Arrays, and Functions

Make use of the structure, variables, and code below to solve the three parts of this problem. Computers are dynamically allocated and tracked using both an array of pointers and a linked list.

```
#define MAX_COMP 42      /* maximum number of computers allowed */

typedef struct computer_t computer_t;
struct computer_t {
    computer_t* next;      /* next computer in list      */
    int         hd_num_MB; /* hard drive size in megabytes */
    int         mem_num_MB; /* memory size in megabytes    */
};

computer_t* computers[MAX_COMP]; /* array of pointers to computers */
computer_t* list_head = NULL;   /* linked list of computers      */
int n_comp = 0;                 /* current number of computers   */
                                /* (in both array and list)     */

int total_memory () {
    int         i;
    int         sum = 0;
    computer_t** computer_ptr = computers;

    for (i = 0; n_comp > i; i++) {
        sum += (*(++computer_ptr)->mem_num_MB);
    }
    return sum;
}
```

Part A (5 points): The function `total_memory()` has an error. Explain the error in fewer than 25 words. *Hint: the syntax is correct.*

Part B (5 points): Rewrite the body of the `for` loop in `total_memory()` to use the `computers` array directly (i.e., without `computer_ptr`).

Problem 3, continued

Part C (10 points): Write a function called `add_computer_to_list` with appropriate parameters that dynamically allocates and initializes a new computer, then adds it to both the `computers` array and **to the end** of the linked list with head `list_head`. Your function should return 1 on success and 0 on failure.

```
int add_computer_to_list (                               )  
{
```

```
}
```

For your convenience, the signatures of the standard memory allocation functions are:

```
void* malloc (size_t size);  
void* calloc (size_t n_elem, size_t elem_size);  
void* realloc (void* ptr, size_t size);
```

Problem 4 (20 Points): Input and Output in C

Part A (7 points): Given the input file shown below on the right, what does the code below print to the screen?

```
int iA=0, iB=0, iC=0, iD=0, iE=0;
char stringA[30], stringB[30];
FILE* f;
```

```
f = fopen("input.txt", "r");
fscanf(f, "%d",&iA);
fscanf(f, "%d%d", &iB, &iC);
fscanf(f, "%s", stringA);
fgets(stringB, 30, f);
fscanf(f, "%d%d", &iD, &iE);
```

```
printf("%d %d %d %d %d\n", iA, iB, iC, iD, iE);
printf("%s\n", stringA);
printf("%s\n", stringB);
```

input.txt

```
1 2
3 z
alpha 4 5
1000
6 a 8
```

Part B (5 points): Explain how `fread()` and `fwrite()` differ from `fscanf()` and `fprintf()`, then give an advantage and a disadvantage of using the former over the latter. For reference, the function declarations are given below.

```
size_t fread(void* ptr, size_t size, size_t nmem, FILE* stream);
size_t fwrite(const void* ptr, size_t size, size_t nmem, FILE* stream);
int fscanf(FILE* stream, const char* format, ...);
int fprintf(FILE* stream, const char* format, ...);
```

Problem 4, continued

Part C (8 points): Write a function to print a 16-bit value in binary, followed by a line feed ('\n'), to the display (stdout). For example, the number 4267 should appear as

0001000010101011

```
void print_binary (unsigned short value) {
```

```
}
```

Problem 5 (20 points): Recursion and Stacks

The five parts of this problem refer to the structure and function shown below.

```
typedef struct node_t node_t;
struct node_t {
    node_t* left;      /* the node's left child      */
    node_t* right;     /* the node's right child     */
    ...                /* other fields (not used in this problem) */
}

unsigned short int func (node_t* root, node_t* target)
{
    unsigned short int subtree;

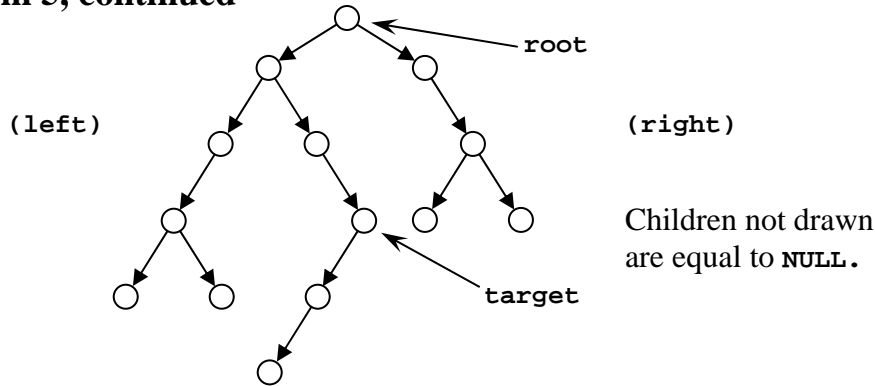
    if (NULL == root) {
        return 0;
    }
    if (root == target) {
        return 0xFFFF;
    }
    subtree = func (root->left, target);
    if (0 != subtree) {
        return (2 * subtree);
    }
    subtree = func (root->right, target);
    if (0 != subtree) {
        return (2 * subtree + 1);
    }
    return 0;
}
```

Part A (4 points): Explain the stopping conditions for the recursion in `func()`.

Hint: answers along the lines of “It stops when root is NULL” will earn no credit.

Part B (2 points): What return values are possible if the `target` parameter is equal to `NULL`?

Problem 5, continued



Part C (6 points): Given the tree shown in the figure above with parameter values as indicated, calculate the return value from `func(root, target)`. Write your answer in either hexadecimal or binary.

Part D (4 points): What does the return value from `func()` represent? Draw a picture showing how it provides such a representation and explain the limitations of the approach used.

Part E (4 points): If this code is compiled for the LC-3, how many memory locations are needed for stack frames (activation records) when processing the call described in **Part C**? Ignore callee-saved registers and any other data not normally associated with the stack frame. **Justify your answer.**

Page 12 Name: _____

Use this page for scratchwork

A.3 The Instruction Set

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD ⁺	0001			DR			SR1			0	00		SR2			
ADD ⁺	0001			DR			SR1			1	imm5					
AND ⁺	0101			DR			SR1			0	00		SR2			
AND ⁺	0101			DR			SR1			1	imm5					
BR	0000			n	z	p	PCoffset9									
JMP	1100			000			BaseR			000000						
JSR	0100			1	PCoffset11											
JSRR	0100			0	00		BaseR			000000						
LD ⁺	0010			DR			PCoffset9									
LDI ⁺	1010			DR			PCoffset9									
LDR ⁺	0110			DR			BaseR			offset6						
LEA ⁺	1110			DR			PCoffset9									
NOT ⁺	1001			DR			SR			111111						
RET	1100			000			111			000000						
RTI	1000			000000000000												
ST	0011			SR			PCoffset9									
STI	1011			SR			PCoffset9									
STR	0111			SR			BaseR			offset6						
TRAP	1111			0000			trapvect8									
reserved	1101															

Figure A.2 Format of the entire LC-3 instruction set. NOTE: + indicates instructions that modify condition codes