

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

**In-Class, Open Book Examination I**  
**March 10, 2011**

The work on this examination is to be your own and you are expected to adhere to the UMass-Boston honor system. All questions can be answered by one or two short sentences. Do not try to make up for a lack of understanding by providing a rambling answer.

**Note: I give partial credit! Show all work!**

**1. (20 points) Short Questions**

a. (2 points) Why is it important to break a long program into smaller functions?

b. (2 points) What is a static variable?

c. (2 points) If I use the list contents of directory command ls, how can I tell the files that are directories?

d. (2 points) What is the difference between command s and n in gdb?

e. (2 points)  $c = (50) ? 10 : 20$ , what is the value of c?

f. (6 points) In C, what is the difference between:

i) 077 and 77?

ii) '\0' and '0'?

iii) "\n" and '\n'?

g. (4 points) Write the binary and 2's complement of 0xfa23196e.

binary: \_\_\_\_\_

2's complement: \_\_\_\_\_

**2. (10 points) Evaluation**

a. (6 points) What values get printed?

```
.....
main(){
    char d[] = "d";
    int a=10, b=10, c;
    c = add2x(a,b,d);
    printf("%2d, %2d, %s", b,c,d);
}
```

```
int add2x(int a, int b, char def[]) {
    int c;
    a += 2;
    b >>= 2;
    c = a + b;
    def[0] = 'g';
    return c;
}
```

\_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_  
b c d

b. (4 points) Expression evaluation:

```
short int d, c = 0x7f00;
c = c + '\x85';
d = c + '\34';
printf("%4x, %6x\n", c, d);
```

/\* what values get printed? \_\_\_\_\_, \_\_\_\_\_ \*/

**CONTINUE ON REVERSE SIDE**

3. (10 points) Indicate how many bytes of memory are allocated in our Sun Sparc machine and what are their initialized values:

	# of bytes:	init. value:
#define LE 80	_____	_____
short int a;	_____	_____
main ( ) {		
static int b;	_____	_____
int array[4];	_____	_____
...		
{		
short int a;	_____	_____
.....		
}		
}		

4. (20 points) After a big wedding celebration, I wrote the following C program. You are asked to find my mistakes (syntax, logic errors). There may be more than 1 error per line (too many cheers!). Identical mistakes in the same line should be counted as one.

Line #	Code	Mistake	Line#	What & Why?
0000	/* main program to compute 6 month sales			
0001	by quarters	1.	_____	_____
0002	enum month{ERR,JAN, FEB,MAR,			
0003	APR, MAY,JUN}	2.	_____	_____
0004	int sales[MAY]={ 100,300,500,200,-200,100};			
0005	int total_sales;	3.	_____	_____
0006				
0007	const banner[]="Total sales=";	4.	_____	_____
0008	main()			
0009	{	5.	_____	_____
0010	int a, c,q1_sales,			
0011	q2_sales =0;	6.	_____	_____
0012	for(c=0; c <= SIX; c++){     /*loop 6 x */			
0013	if(c= JAN   c= FEB   c=MAR)	7.	_____	_____
0014	q1_sales += sales(c);			
0015	else	8.	_____	_____
0016	q2_sales += sales(c);			
0017	}	9.	_____	_____
0018	total_sales += q1_sales + q2_sales;			
0019	banner[11] = ':';	10.	_____	_____
0020	printf( "%s =%s", banner, total_sales);			
0021	}			

5. (40 points) Word spotter Program

Write a C program to:

1. Read an input file (via redirection of the standard input).
2. Find out how many times the word UMASS appear.
3. This continues until you reach the EOF.
4. Find out how many total words you have read in.
5. Print out the count of the word UMASS and the total word count.

\*\*\* Do not use other library functions besides printf(), getchar(), getline(), putchar(). If you use the getline() function from the book, you do not have to write out the code.

Show your pseudocode and C code.

Answers:

1.
  - a. For better readability.
  - b. A local variable that retains its values between calls.
  - c. `ls -al` will show all the files and subdirectories within the directory. The entries that start with `d` are the subdirectories
  - d. `gdb` command `s` means to execute next line  
`gdb` command `n` means to execute next line. If the next line calls a function, it will execute the entire function call
  - e. `c=10`
  - f. `077` =octal 77 =decimal 63;  
77 is decimal 77  
`'\0' = 0 ;`  
`'0' = ascii value of 0 =0x30`  
`"\n" = {'\n', '\0'};`  
`'\n' = {'\n'}`

- g. 1111 1010 0010 0011 0001 1001 0110 1110  
0000 0101 1101 1100 1110 0110 1001 0010

2.
  - a. `b=10 c=14 d=g`
  - b.

```
0111 1111 0000 0000
+ 1111 1111 1000 0101
0111 1110 1000 0101  c=0x7e85
+ 0000 0000 0001 1100
0111 1110 1010 0001  d=0x7ea1
```

3.

0	literal value
2	0
4	0
16	garbage
2	garbage

4.
  - 001 → missing `*/` for the comment line
  - 000 → missing `#include <stdio.h>`
  - 003 → missing `;`
  - 004 → `MAY=5`, `sales[]` is initialized w/6 members
  - 007 → need a data type; `const char banner[]...`
  - 011 → `q2_sales` is not defined

0012 → `SIX` not defined  
0012 → logic error: `c < SIX` if `SIX=6`  
0013 → should be `c==` instead of `c=`  
0013 → logic error `c==JAN-1`, `c==FEB-1`, etc  
0013 → should be `||` instead of `|`  
0014 → logic error; `q1_sales` not initialized  
0014 and 0016 → should be `sales[c]`  
0019 → cannot modify a const string  
0021 → should be `%d` instead of `%s` for `total_sales`

5.  
`/* Word Spotter and counter pseudo Code begins here`

initialize variables  
loop until the length of the read-in line is  $\leq 0$

Go through the entire line and look for space, or a tab, or a null character or a new line character,

If it is between, increment word counter

Go through the entire line and look for a space, or a tab or a new line character or a null character,

if it is, check the previous 5 characters ==  
`'U','M','A','S','S'`

if they are, increment the match counter

end of loop

print out the match count and word count

`*/`

5. (cont'd)

```
/* Word Spotter Program */
#include <stdio.h>
#define MAXLINE 1000
int getline1(char line[], int maxline);

int main(void)
{
    int i, len, middle, word_c=0, count=0;
    char line[1000];

    /* read in a line */
    while((len = getline1(line, MAXLINE)) > 0){

        /* code to count words */
        middle = 0;
        for (i = 0; i < len; i++){
            if( (line[i] == ' ' || line[i] == '\t' || line[i] == '\0' || line[i] == '\n') && middle == 0) continue;
            middle = 1;
            if(line[i] == ' ' || line[i] == '\t' || line[i] == '\0' || line[i] == '\n'){
                middle=0;
                word_c++;
            }
        }

        /* code to count UMASS */
        for (i=0; i < len; i++){ /* check for space, new line, tab */
            if( i > 4 && (line[i] == ' ' || line[i] == '\t' || line[i] == '\0' || line[i] == '\n')){
                if(line[i-1] == 'S' && line[i-2] == 'S' && line[i-3] == 'A' && line[i-4] == 'M'
                   && line[i-5] == 'U') /* if it is, check the last 5 characters */
                    count++; /*if it is 'U','M','A','S','S', increment counter */
            }
        }
    }
    printf( "number of times UMASS occurred =%d in %d words\n", count, word_c);
}

int getline1(char s[], int lim){
    int c, i;
    for (i=0; i<lim-1 && (c=getchar()) !=EOF && c !='\n'; ++i) s[i] =c;
    if(c=='\n'){
        s[i] =c;
        ++i;
    }
    s[i]='\0';
    return i;
}
```