**Operating Systems Pre-test**

Since the first and second year courses changed dramatically with your year, I want to see where you stand on some topics. This test does not count for your grade.

* Please do at least one question per topic.
* If you do not know how to answer any questions in a particular topic, please write – *I do not know this* - next to it.
* If you’ve learned it but forgotten it, just please give it a try.

**Topic 1: Memory and pointers**

**Question 1**

* Answer the two questions in the code.

int \*f2( int a, int \*ptr )

{

int b;

b = \*ptr + a;

// **If the program is executing, what’s the value of b right here?**

\*ptr = b;

return( &b );

}

int main()

{

int x;

int \*p, \*q;

x = 6;

p = &x;

q = p;

\*q = 10;

q = (int \*) malloc( sizeof(int) );

\*q = 4;

p = f2( x, q );

**// Explain what is wrong with the following statement**

**// use precise technical language**

x = \*p;

}

**Question 2 (Memory and pointers)**

* The following code compiles with no errors or warnings (header files not shown for space). Answer the questions after the code.

int main()

{

char \*sentences;

int a;

sentences = (char \*) malloc(sizeof(char)\*10);

for (a = 0; a < 12; a++){

sentences[a] = 'b' + a;

}

for (a = 0; a < 12; a++){

printf( "%c\n", sentences[a] );

}

printf("\n");

sentences = (char \*) malloc(sizeof(char) \* 5);

for (a = 0; a < 5; a++){

sentences[a] = 'l' + a;

}

for (a = 0; a < 10; a++){

printf( "%c\n", sentences[a]);

}

free(sentences);

}

* There are several problems in this code. List as many as you can find (or identify them in the code).
* What will this program print out?
* Where is a memory leak in this program? Explain.
* What is freed with the free() command at the end of the code?

**Question 3 (Memory and pointers)**

* Identify and describe the **2** memory (runtime) errors in the program below.

void foo( int a, char \*\*b )

{

\*b = (char \*) malloc( ++a );

strcpy( \*b, "bye" );

}

int main()

{

int x = 3;

char str[3];

char \*p;

p = (char \*) malloc( 3 );

strcpy( p, "ok" );

strcpy( str, "bye" );

foo( x, &p );

// program continues …

}

**Topic: C programming**

**Question 1**

* Examine the following code which reads lines from a file into an array of strings.
* At the end of the existing code, write a loop that traverses the array, and for each string in the array, calls a function with the signature

void reverse ( char \*str );

Also write the function reverse, which reverses its string argument (e.g., “what?” becomes “?tahw” )

* Following the loop written above, write a second loop that sorts the array, in place, by the length of each string (longest to shortest).
* Write a third loop that prints each string, and then frees the memory for it.

#define MAX\_LINES 10

#define BUFSZ 128

int main()

{

int i = 0, count = 0;

char \*lines[MAX\_LINES];

char buf[BUFSZ];

FILE \*fp;

if ( ( fp = fopen( "something.txt", "rw" )) == NULL )

exit( -1 );

while ( fgets( buf, BUFSZ-1, fp ) != NULL && i < MAX\_LINES )

{

lines[i] = (char \*) malloc( strlen(buf) + 1 );

strcpy( lines[i], buf );

i++;

count++;

}

**// Your loops go here**

**Question 2 (C Programming)**

* Look at the following code and answer the questions following it.

int main( int argc, char \*\*argv )

{

int \*xyz, i;

if (argc < 3)

exit(-1);

xyz = // **FILL IN THE REST OF LINE**

for ( i = 1; i < argc; i++ )

{

\*(xyz+i-1) = atoi(argv[i]);

**// REWRITE LINE ABOVE USING ARRAY SYNTAX**

}

printf( "%d %c\n", \*(xyz+1), \*(argv[1]+1) );

// program continues ...

}

* In the space above, add the most appropriate missing code to initialize **xyz** so that the code will work.
* In the space above, rewrite the line inside the **for** loop using more readable array syntax, rather the than pointer syntax used there.
* Suppose the name of this program is **a.out**, what will print if we type the following at the bash prompt:

% ./a.out 45

% ./a.out 45 78 99

* Explain the difference between what **+1** means in each place it is used in the printf call from the program above, as shown below:

printf( "%d %c\n", \*(xyz**+1**), \*(argv[1]**+1**) );

**Topic: Unix command line**

**Question 1**

* Suppose I am working in a terminal window and my current working directory is my home directory /home/mrizvi and I ***successfully*** compile my program test.c using the command:

gcc src/test.c -o myprog

If I immediately execute the ls command, **list what will minimally be printed**.

**Question 2 (Unix command line)**

**Explain** what the following commands do.

* chmod +x myfile

* ls ../src/\*.java

* cat < Afile > Bfile
* ls New Folder