Problem 1 (3 points, 5 minutes)

What is printed by the following C program segment, run on the EECS instructional computers? (The notation 0x... specifies a hexadecimal constant; in a printf format string, %x specifies output in base 16.)

```
int *ptr;
ptr = 0x1050;
printf ("%x\n", ptr--);
printf ("%x\n", ptr);
```

Problem 2 (5 points, 15 minutes)

Write a C function named resultOfInsert that, given a string s, a character c, and a position k between 0 and strlen(s), inclusive, returns the string that results from inserting c into s at position k. For example,

```
resultOfInsert ("abcd", '_', 2)
```

should return the string "ab_cd". The insertion should not change the argument string. You may use any functions declared in <string.h>. You don't need to do any error checking.

```
char *resultOfInsert (char *s, char c, int k) {
```

Problem 3 (3 points, 9 minutes)

Consider the following C program.

```
#include <string.h>

int main ( ) {
   char oneName[10] = "xxxxxxxxx";
   char* anotherName;

   anotherName = oneName;
   strcpy (anotherName, oneName);
   oneName[1] = 'O';
   return 0;
}
```

Draw the box-and-pointer diagram that represents the oneName and anotherName arrays and their contents after executing this segment.

Problem 4 (5 points, 15 minutes)

Consider the storage layout below, which uses the boundary tags system from lab assignment 3.

"addresses"	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
contents	3	0	0	-2	3	-4	-7	-7	-7	-7	-4	-3	-13	-13	-13	-3	2	0	0	2

Part a

The layout provides enough information to determine the storage blocks represented, along with which blocks are allocated and which blocks are free. Provide this information below by specifying the start and end address of each block, including the boundary tag information. (For example, at the start of the program there is one free block that starts at "address" 1 and ends at "address" 20.)

start "address"	*end "address"*	*allocated or free?*

Part b

The layout above is incorrectly structured. What's wrong with it?

Problem 5 (3 points, 5 minutes)

Suppose that the label names marks the beginning of an array of strings. In MIPS assembly language, this might appear as follows:

```
names:   .word    starting address of first string
         .word    starting address of second string
          ...
```

Give a MIPS assembly language program segment that loads the fourth character of the second string into register $t0. For example, if the array contains the strings "mike", "clancy", "dave", and "patterson", this character would be the 'n' in "clancy". Assume that there are at least two strings in the array and at least four characters in the second string.

A three-line solution is sufficient. You may use any registers you want.