cs61c, Spring 1998
Midterm #1
Professor Clancy

Problem #1 (7 points, 14 minutes)

In lab assignment 4, you wrote a function that returned the contents of the various fields of a MIPS I-format instructio
of the bits is as follows.

```
     The first 8 bits are the op code.
     The next 4 bits are the register to be modified by the instruction.
     The last 20 bits are the immediate operand, in 1's complement.

Thus the equivalent to the MAL instruction, addi $10,-2 might appear in hexadec
      94 af ff fd
if the op code for the addi instruction were 94 base 16.
```

On the next page, write a MAL function SplitIFormat that returns the contents of the register and immediate fields of a

```
        void SplitIFormat (int instr, int *register, int *immediate);

If written in C++, its prototype would be
        void SplitIFormat (int instr, int &register, int &immediate);
```

Follow the conventions described in class and in lab and homework assignment 6 for passing arguments and managing

Problem #2 (7 points, 13 minutes)

Part a Translate the following MAL program segment to TAL. You may use either names or numbers for the registers.

```
                li      $t1,-5
        loop:   sub     $t1,$t1,3
                bgt     $t1,$a1,loop

Equivalent TAL segment:
```

Part b

Your answer to part a should include a branch instruction. Translate this branch instruction to machine language by fil

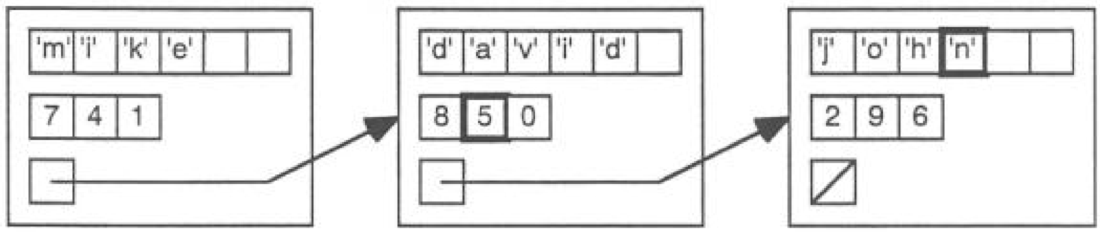
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Problem #3 (7 points, 10 minutes)

Consider a list with nodes defined in C or C++ as follows.

```
        struct ListNode {
                char name[6];
                int code[3];
                struct ListNode* next;     /* ListNode* next in C++ */
        };
```

The diagram below, not drawn to scale, gives an example of such a list.



Part a

Assume that register $a1 contains a pointer to the first node of the list. Write MAL code that loads $s2 with the second

Part b

Again, assume that register $a1 contains a points to the first node of the list. Write MAL code that loads $s2 with the f

Problem #4 (7 points, 12 minutes)

Consider the following C functions that check if one string contains another as

```
        int ContainsAsSubstring (char *s1, char *s2) {
                if (*s2 == '\0') {                /* if string 2 has run out, */
                        return 1;                 /* it's a substring of string 1
                } else if (*s1 == '\0') {         /* if string 1 has run out, */
                        return 0;                 /* string 2 isn't a substring o
                } else if (StartsWith (s1, s2)) {
                        return 1;
                } else {
                        return ContainsAsSubstring (s1+1, s2);
                }
        }
        int StartsWith (char *s1, char *s2) {
                if (*s2 == '\0') {                /* any string starts with the e
                        return 1;
                } else if (*s1== '\0') {          /* if string 1 has run out, */
                        return 0;
                } else if (*s1 != *s2) {
                        return 0;
                } else {
                        return ContainsAsString (s1+1, s2+1);
                }
        }
```

Some examples of how ContainsAsSubstring behaves are listed below.

String 1	String 2	result of ContainsAsSubstring
"abcde"	"abc"	1
"xyabc"	"abc"	1
"axbc"	"ab"	0
"xy"	"abc"	0

Fill in the missing code in the MAL implementation of ContainsAsSubstring below. (Don't worry about StartsWith.) Y registers and the system stack. You may assume that neither argument pointer is null.

```
ContainsAsSubstring:
        # save registers on the stack






        #check base cases




        beqz  $t1,returnTrue
        beqz  $t0,returnFalse

        move  $s0,$a0           #does string 1 start with string 2?
        move $s1,$a1
        jal   StartsWith
        bnez  $v0,returnTrue

        add    $a0,$s0,1                 #no match; make recursive call
        move $a1,$s1
        jal     ContainsAsSubstring
        j        return

returnTrue:
        #prepare to return 1



        j     return
returnFalse:
        #prepare to return 0




return:
        # restore registers and return
```

Problem #4 (7 points, 12 minutes)