Name:____

University of California, San Diego
CSE 30 – Computer Organization and Systems Programming
Fall 2010 – Final
Prof. Ryan Kastner

Name	
Student ID	

Name of person to your left	
Name of person to your right	

You will lose points if you do not put your name on the top of every page and complete the all of the above information.

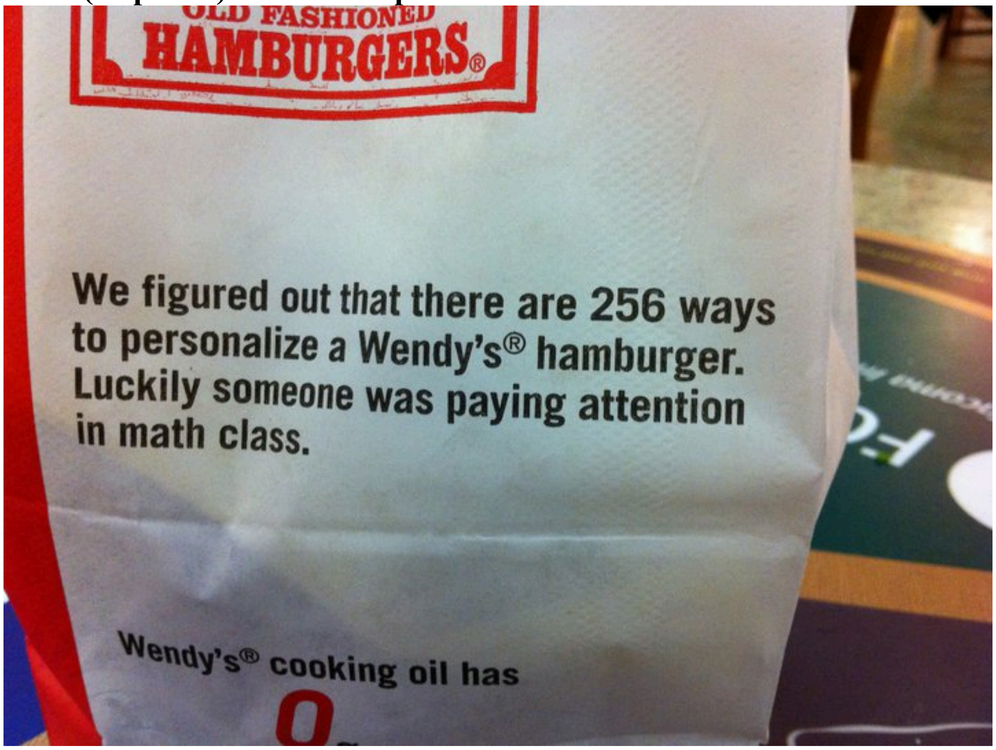
Problem	Possible	Score
1	20	
2	20	
3	20	
4	40	
5	35	
6	15	
Total	150	

Name:____

This page was intentionally left blank.

Name:____

Problem 1: (20 points) Number Representation



(5 points) How many yes/no options (e.g., mustard/no mustard, lettuce/no lettuce, etc.) do you have to "personalize" your Wendy's hamburger?

(5 points) If Wendy's added two additional yes/no options, how many total ways would one now have to "personalize" your Wendy's burger? Assume that these two options are totally independent of the 256 current ways of personalization.

(5 points) Assume that In-N-Out has N ways to personalize their burger. How many bits do they require to represent all of these personalization options?

(5 points) Knowing the secret In-N-Out menu expands the number of options by 17 times, i.e., there are 17N ways to personalize your burger if you know the secret menu options. How many bits does this require?

Name:____

Problem 2: (20 points) Arithmetic

Translate the following code into MIPS assembly:

```
Z = ((A+B)*(C+D))+((E+F)*(G*H))
```

Assume that variables `A-H,Z` are in registers `$s0-$s8`, respectively (e.g., `C = $s2`). Also, assume that you cannot overwrite variables `A-H` since they will all be used later in the program. You must write the MIPS code exactly as it appears above, i.e., you can not perform any optimizations (e.g., using transitive, commutative, distributive properties).

For this question, you can use the `mul` pseudo-instruction. It has this form:

`mul $1, $2, $3`, where `$1 = $2 * $3`.

It does the same thing as:

```
mult $2, $3
mflo $1
```

Do not use mult and mflo.

a) **(10 points)** Assume that you have sequential processor where `add` and `mul` take 1 and 3 cycles, respectively. Write the code such that it uses the minimum number of registers and cycles. How many additional registers (other than `$s0-$s8`) does your code require? How many cycles does your code need?

Name:____

b) **(10 points)** Assume that you have a different type of processor architecture (call it VLIW) that can perform one `add` operation and one `mul` operation during every cycle. Rewrite the code to take advantage of this and use the <u>minimum</u> number of cycles. The VLIW processor uses a faster implementation of `mul`, which only takes 1 cycle, the same as the cycle time of `add`. To make it easier to grade, please write `mul` and add that are executed in the same cycle on one line, for example:

```
mul $1, $2, $3,     add $4, $2, $8
```

How many cycles does your code need?

Name:____

Problem 3: (20 points) String Manipulation

(5 points) Describe succinctly, in English, what the following function does:

```
char * someFunction(char *src, char uc, int n)
 {
     while (n-- != 0) {
          if (*src == uc)
               return src;
          src++;
     }
     return NULL;
 }
```

(15 points) Write the C code for the function
`char * insert (int pos, char * base, char * insert)`

that inserts a copy of the entire contents of `insert` into `base` at character position `pos`.

```
main()
{
     char * base = "CSE Computer Organization";
     char * insert = "30";
     char * result = insert(3, base, insert);
}
```

The result string is "`CSE30 Computer Organization`".

Name:____

This page was intentional left blank.

Name:____

Problem 4: (40 points) MIPS Reverse Compilation

Consider the following MIPS assembly code:

```
creeper:        add $t0, $0, $0
                addi $t1, $a1, -1
specter:        slt $t2, $t0, $t1
                beq $t2, $0, disc_demon
                add $t3, $a0, $t0
                lb $t4, 0($t3)
                add $t5, $a0, $t1
                lb $t6, 0($t5)
                sb $t6, 0($t3)
                sb $t4, 0($t5)
                addi $t0, $t0, 1
                addi $t1, $t1, -1
                j specter
disc_demon:     add $v0, $a0, 0
                jr $ra
```

a) **(15 points)** Translate the above `creeper` function into C. Your function header should list the types of any arguments and return values. Also, your code should be as concise as possible, without any gotos. We will not deduct points for syntax errors unless they are significant enough to alter the meaning of your code.

b) **(5 points)** Describe briefly, in English, what this function does.

Name:____

c) **(20 points)** Convert the following instructions from the code above into 32 bit hexadecimal number. Assume that the address of the first instruction (`add $t0, $0, $0`) or equivalently the label `creeper` is located at address `0x00400018`.

(5 points) `lb $t6, 0($t5)`

(5 points) `beq $t2, $0, disc_demon`

(5 points) `j specter`

(5 points) `addi $t1, $a1, -1`

Problem 5: (35 points) Data Structures

A list is a series of elements that are sequentially connected to each other. A list node is a structure that represents a single element of a list. A list node is formally defined as follows:

```
struct ListNode {
      int         data;  // this is the data contained in this element
      ListNode*   next;  // this is a pointer to the next element in the list
      ListNode*   prev;  // this is a pointer to the previous element in the list
};
```

A list is simply a series of these nodes connected using pointers.

The C function `reverse` that completely reverses the ordering of the elements in the list is shown below. The function takes one argument: a pointer to the first `ListNode` object of the list. It returns a pointer to the new first element of the list (which was previously the last element)

```
ListNode * reverse(ListNode *aNode)
{
      ListNode * tempNode;

      tempNode = aNode->next;     //save the next pointer
      aNode->next = aNode->prev;  //switch the prev and next ptrs
      aNode->prev = tempNode;

      if(aNode->prev == NULL)     //stop if end of list
           { return aNode; }      //    has been reached
      else
           { return reverse(aNode->prev); } //otherwise continue
                                            //    recursively
}
```

a) **(5 points)** How many bytes is one instance of a ListNode struct?

b) **(20 points)** Write a *recursive* MIPS assembly code for this C function. You must follow register conventions as well as standard procedure calling conventions for full credit on this question. In other words, make no assumptions about the calling procedure.

- Solutions that are not recursive will not get any credit
- You must follow all register conventions and procedure calling conventions
- You must only use real MIPS instructions (no pseudoinstructions)
- You must write comments. Code that is not adequately commented will be penalized.

Name:____

c) **(10 points)** Translate that following memory layout to the pictorial description of a list. The head of the list is `0x472AF014`.

Address	Value	Address	Value
0xFFFFFFFF	. . .		. . .
	0x00000000		0xE123014C
	0x472AF014		0x00123AF0
	0x472AF014		0xE123014C
	0x123ABC00		0x00123AFO
0xE123014C	0x80042AB0	0x123ABC00	0x041ABC28
	. . .		. . .
	0x041ABC28		0x472AF014
	0x00123AF0		0x80042AB0
	0x00000000		0x00000000
	0x00000000		0x00000000
0x80042AB0	0x00000003	0x041ABC28	0x00000004
	. . .		. . .
	0xE123014C		0x80042AB0
	0x041ABC28		0x00000000
	0x00123AF0		0x123ABC00
	0xE123014C		0x472AF014
0x472AF014	0x00123AF0	0x00123AF0	0xE123014C
	. . .		. . .

Name:____

Problem 6: (15 points) Handle Swap

```
void swap(char **one, char **two) {
     char temp = **one;
     **one = **two;
     **two = temp;
}
```

Write MIPS assembly code for the `swap` C function.

- You must follow all register conventions and procedure calling conventions
- You must only use real MIPS instructions (no pseudoinstructions)
- You must write comments. Code that is not adequately commented will be penalized.