University of California, Berkeley – College of Engineering
Department of Electrical Engineering and Computer Science

Fall 2004 | Instructor: Dr. Dan Garcia | 2004-10-18

☺CS61C Midterm☺

Last Name	
First Name	
Student ID Number	
Login	
The name of your TA (please circle)	
Name of the person to your Left	
Name of the person to your Right	
All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61C who have not taken it yet. ***(please sign)***	

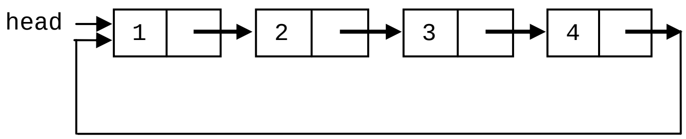
Instructions

- This booklet contains 6 numbered pages including the cover page. Put all answers on this pages, don't hand in any stray pieces of paper.
- Please turn off all pagers, cell phones & beepers. Remove all hats & headphones. Place your backpacks, laptops and jackets at the front. Sit in every other seat. Nothing may be placed in the "no fly zone" spare seat/desk between students.
- Question 0 (1 point) involves filling in the front of this page and putting your name & login on every front sheet of paper.
- You have 180 minutes to complete this exam. The times listed by each problem will allow you to finish with 60 (!) minutes left to check your answers. The exam is closed book, no computers, PDAs or calculators. You may use one page (US Letter, front and back) of notes.
- There may be partial credit for incomplete answers; write as much of the solution as you can. We will deduct points if your solution is far more complicated than necessary. When we provide a blank, please fit your answer within the space provided. You have 3 hours…relax.

Problem	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Total**
Minutes		30	5	5	10	20	10	20	20	120
Points	1	22	6	3	4	16	5	10	8	75
Score										

Question 1: C and Circular Lists (22 points – 30 min.)

We're writing a circular linked list to keep numbers. The idea is very similar to a single-linked list, but the last element points to the first. Our circular linked list is made up of elements of type `pair` (a data type from CS61A and project 1). Assume when the list is empty we initialize the *global variable* `head` to `NULL`. Here's an example on the left, with the `pair` definition on the right:



The `pair` structure is defined as follows:

```
struct pair {
    int car;           // "number"
    struct pair *cdr; // next "pair"
} *head;
```

In the figure above, we can see 4 elements linked. When we insert an element, it goes *after the first element****.*** E.g., if we represent the distinct elements list in the example above *before* insertion as `{1 2 3 4}`, then *after* a call to `insert(5)` it would be `{1 5 2 3 4}`.

a) Help us to write the `insert` function by adding only 3 statements.

```
void insert(int d) {
      /* create the new node */

      tmp->car = d;

      /* Insert the new node in the right place */
      if ( head == NULL ) {
            /* The struct was empty… link the itself and we're done */
            tmp->cdr  = tmp;
            head      = tmp;
      } else {
            /* There were already elements in the linked list.
                Link the new node after the first element. */


      }
}
```

b) Instead we'd like you to link it after the "second" element. (If we only have 1 element, do the same as before. Can it be done by only modifying the 2nd and 3rd statements? If so, do it below. If not, explain why.

c)

```
/* Link the new node after the "second" element */
```

Or

Question 1 (continued): C and Circular Lists (22 points – 30 min.)

d) Now, we want to be able to delete the full structure. Assume that the OS immediately fills any freed space with garbage, so you cannot access freed heap contents. Finish the recursive `delete_recursively` function. We want the tightest, cleanest code possible (measured by the number of statements which terminate in semicolons). If you use only *2 semicolons*, full credit. If you use 3, you'll lose 1 point. If you use more, you'll lose 2 points.

```
void delete_full_structure()
{
	if (head == NULL )
		return;

	delete_recursively(head);
}
```

You saw how inserting a fifth element numbered 5 into our list messed up our numbering. We'd like to write `reset_numbers` that *clobbers* the node numbers to restore the nice `1, 2, 3,`… numbering. Note: A pointer to a `struct` stored in memory is just a pointer to memory we treat as broken up into the fields.

```
	void reset_numbers(pair* p, int i)
	{
		if (p != NULL ) {
			p->car = i;
			reset_numbers(->cdr, i++);
		}
	}
```

e) Convert `reset_numbers` to MIPS **keeping its structure recursive**. I.e., don't hand-optimize.

prologue

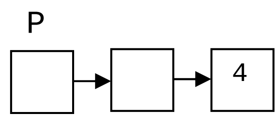
body

epilogue

f) In one sentence, what happens *on an actual MIPS machine* if we call `reset_numbers(head, 1)` on the lists as described in this problem? (Assume our list is not empty).

Question 2: malloc masters (6 points – 5 min.)

Fill in the following C function that creates and returns the pointer diagram below. You can use only one variable in your code, p. Don't use more than 5 statements, *including* the two we show below. You may modify line 1 (between `int` and p), but you're not allowed to touch line 5. Please specify the function's return type.



Structure returned by `malloc_masters()`

```
int** malloc_masters () {

      int ** p;                                   /* 1st statement */
_____________________________________________________________________

_____________________________________________________________________

_____________________________________________________________________

      return p;                                   /* 5th statement */
}
```

Question 3: Compiling, Assembling, Linking, Loading (3 points, 5 min.)

In at most one sentence each, describe 2 advantages and 1 disadvantage of *dynamically-linked* (vs statically-linked) *programs*.

Advantage #1:	
Advantage #2:	
Disadvantage:	

Question 4: Raw Bits (4 points, 10 min.)

		0x02556321			
Interpret the word on the right as…	four characters				
	an instruction				

Question 5: MIPS Reverse Engineering (16 points – 20 min.)

a) Translate the following MIPS function into the C in the boxes on the right. Fill in the arguments (& their data types) and return types for `foo` & `bar`.

```
Main: …
      ## Set up $a0
      jal foo
      …

foo:  li    $a1, 0
bar:  addi  $sp, $sp, -4
      sw    $ra, 0($sp)
      bnez  $a0, else
      mv    $v0, $a1
      j     end
else: srl   $a0, $a0, 1
      addi  $a1, $a1, 1
      jal   bar
end:  lw    $ra, 0($sp)
      addi  $sp, $sp, 4
      jr         $ra
```

If you can tighten the body of `bar` to be just "`return ___;`" (it's possible) you'll receive full credit. Otherwise you'll lose one point.

b) What math function does `foo` compute?

c) What's the biggest number that `foo` will ever return?

d) What does `foo((unsigned int) –x)` return if `x` is a single-digit integer [1,9]?

Question 6: Binary Encoding (5 points, 10 min.)

a) How many different instructions can we specify in MIPS given our standard 32-bit encoding? Assume we only have R-, I- & J-format instructions.

b) (This question has nothing to do with MIPS) Assume we have enough bits to byte-address 16_{10} exbibytes. We want to define some number of the most-significant bits to encode $12_{10} \times 2^{10}$ things, and some number of the least-significant bits to encode $200{,}000_{10}$ things. How many things can we encode with the remaining bits? Use IEC language, like "16 kibithings," or "128 mebithings." Show your work below.

Question 7: Numerical Representation (10 points – 20 min.)

Considering *8-bit integers*, answer the following questions for each column. The bits are numbered as: `7 6 5 4 3 2 1 0`. Each box might be a different integer. You *must* show scratch work to receive credit.

<table>
<tr><th colspan="2"></th><th colspan="2">Given that bits 3-0 are 1111</th><th colspan="2">Given that bits 7-4 are 1001</th></tr>
<tr><td colspan="2">If the # were interpreted as a two's complement signed integer, would it be negative (-), positive (+) or impossible to tell? (circle one)</td><td colspan="2">- CAN'T-TELL +</td><td colspan="2">- CAN'T-TELL +</td></tr>
<tr><td rowspan="12">If the # were interpreted as a [each of the values on the right], what is the most negative (closest to -∞) value possible?

(for each answer, show your work and write the decimal and hexadecimal value immediately below)</td><td rowspan="3">Sign-magnitude</td><td colspan="2">Scratch space</td><td colspan="2">Scratch space</td></tr>
<tr><td>Decimal Value</td><td>Hexadecimal Value</td><td>Decimal Value</td><td>Hexadecimal Value</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="3">Unsigned</td><td colspan="2">Scratch space</td><td colspan="2">Scratch space</td></tr>
<tr><td>Decimal Value</td><td>Hexadecimal Value</td><td>Decimal Value</td><td>Hexadecimal Value</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="3">Two's complement signed</td><td colspan="2">Scratch space</td><td colspan="2">Scratch space</td></tr>
<tr><td>Decimal Value</td><td>Hexadecimal Value</td><td>Decimal Value</td><td>Hexadecimal Value</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="3">One's complement</td><td colspan="2">Scratch space</td><td colspan="2">Scratch space</td></tr>
<tr><td>Decimal Value</td><td>Hexadecimal Value</td><td>Decimal Value</td><td>Hexadecimal Value</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>

Question 8: Floating Point Debate (8 points – 20 min.)

Bush and Kerry are debating about which is better for representing integers with 32 bits, a `float` or an `int`; you're going to provide them with data to support their argument. We define two acronyms here: **NICTO = "Negative Integer Closest To 0"** and **PICTO = "Positive Integer Closest To 0."**

a) What are the **NICTO** and **PICTO** that float *can* represent but int *cannot*.
Show all work and the 32-bit hex number that corresponds to both.

Show all of your work	**NICTO**	**PICTO**
Decimal Value (you may leave as an expression)		
32-bit Hexadecimal Value		

b) What are the **NICTO** and **PICTO** that int *can* represent but float *cannot*. Show all work.

Show all of your work	**NICTO**	**PICTO**
Decimal Value (you may leave as an expression)		