CS 60B Midterm #2 — April 5, 1993

Your name ______________________________

login c60b–_______

Discussion section number _______

TA's name ______________________________

This exam is worth 15 points, or 15% of your total course grade. The exam contains four substantive questions, plus the following:

Question 0 (1 point): Fill out this front page correctly and put your name and login correctly at the top of each of the following pages.

This booklet contains five numbered pages including the cover page. Put all answers on these pages, please; don't hand in stray pieces of paper. This is an open book exam.

When writing procedures, write straightforward code. Do not try to make your program slightly more efficient at the cost of making it impossible to read and understand.

When writing procedures, don't put in error checks. Assume that you will be given arguments of the correct type.

Our expectation is that many of you will not complete one or two of these questions. If you find one question especially difficult, leave it for later; start with the ones you find easier.

0	/1
1	/3
2	/3
3	/4
4	/4
total	/15

Question 1 (3 points):

Without using any global variables, write a C function `counter` that takes no arguments, and returns an integer equal to the number of times it's been invoked. That is, it should return `1` the first time, `2` the second time, and so on.

Your name ____ login c60b–_______

Question 2 (3 points):

Consider the following C function to multiply two rational numbers:

```
struct rational {
    int num, den;
};

typedef struct rational Rat;

Rat mul(Rat x, Rat y) {
    Rat r;

    r.num = x.num * y.num;
    r.den = x.den * y.den;
    return r;
}
```

Several statements about this function are shown below. Some are true and some are false. Circle the true ones.

(a) The program is incorrect but would work if the declaration of `r` were changed to say

```
    Rat r = (Rat)malloc(sizeof(struct rational));
```

and the `typedef` were changed to say `struct rational *Rat` (adding the asterisk).

(b) The program is incorrect but would work if the variable `r` were declared as `static`.

(c) The program is correct as shown.

(d) The storage for variable `x` is in `mul`'s stack frame.

(e) The storage for variable `r` is in `mul`'s stack frame.

(f) The variable `x` fits in a register.

Question 3 (4 points):

Consider the following C procedures:

```
void FirstToEnd(char a[]) {
    int i;

    for (i=1; a[i] != '\0'; i++)
        swap(&a[i-1], &a[i]);
}

void swap(char *x, char *y) {
    char temp;

    temp = *x;
    *x = *y;
    *y = temp;
}
```

Procedure `FirstToEnd` modifies a character string by rotating the first character to the end of the string, so that if the original string is `"garply"` then it will be changed to `"arplyg"`. This is not the most efficient possible implementation! We designed it as an easy example that uses `swap` as a subprocedure.

Your job is to translate `FirstToEnd` into MIPS assembler. You may assume that someone else has translated `swap`, using the standard MIPS conventions (with arguments in registers). Don't do the swapping yourself within `FirstToEnd`! You must use the provided `swap` procedure.

Your name ____ login c60b–_______

Question 4 (5 points):

We are implementing linked lists of integers using the following definitions:

```
struct node {
    int value;
    struct node *next;
};

typedef struct node *List;

#define NIL (List)0
```

Write a C function `is_ordered` that takes a list as argument and returns an integer: `1` if all of the elements of the list are in increasing order (that is, each element is greater than or equal to the one before it), or `0` if any element is out of order.