CS 60B Final — December 14, 1992

Your name _____

login c60b–_____

Discussion section number _____

TA's name _____

This exam is worth 24 points, or 24% of your total course grade. The exam contains six questions.

This booklet contains eight 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, 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.

1	/3
2	/3
3	/4
4	/4
5	/4
6	/6
total	/24

Question 1 (3 points):

(a) Convert `0xffffff99` to a signed decimal integer.

(b) Suppose we want to represent positive integers using a *binary coded decimal* representation, with each decimal digit represented in four bits, so that `123` would be represented as `0001 0010 0011`. With this representation, what is the largest number that fits in a MIPS word?

(c) Of the MIPS assember instructions below, **circle** the ones that **can** be carried out by a single MIPS hardware instruction:

```
        la $8, 0x80000080

        move $10, $4

loop:   beq $8, $9, loop

loop:   blt $8, $9, loop

        la $8, 0x2000
```

Your name _____ login c60b–_____

Question 2 (3 points):

You have been hired to design a 64-bit version of the MIPS architecture. You're deciding between two possibilities for instruction formats: Option A: Double the width of every field in the existing instruction formats (12 bits for opcode, 10 bits for registers, etc.). Option B: Keep all fields the same as in the 32-bit version except for the immediate/offset field, which will be 48 bits wide.

Give an argument (one or two sentences should be enough) for why each of these choices might make programs run faster. (Don't just say, for example, "it can address more registers"; explain why that will make programs *faster*.)

Which argument seems more reasonable to you, and why? (We haven't chosen a right answer to this; it's your reason we want to see.)

Question 3 (4 points): Translate the following fragment of C code into MIPS assembler:

```
int baz(int i, int a[7], char b[10]) {
    a[i] = b[i];
    return a[0];
}
```

Your name _____ login c60b–_____

Question 4 (4 points):

Write a C function satisfying this declaration:

```
char *doubles(char *str);
```

that returns a string containing all characters that appear twice (or more) in a row in the argument string. For example,

```
printf("%s\n", doubles("Mississippi"));
```

should print "`ssp`". The result should be returned in a newly allocated string that's no bigger than necessary. You may assume that the argument string has fewer than 100 characters.

Question 5 (4 points):

You are working with lists of integers constructed out of nodes like this:

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

Write the C procedure `swap` that takes a pointer to a list as its argument. It should return a list with the same elements, but reversing the order of each pair of elements. For example, if the list that you get as the argument contains the elements `(22 18 9 5 47 100 6)` then the returned list should contain `(18 22 5 9 100 47 6)`. (Note that if the list has an odd number of elements you leave the last one alone.) Do the reordering without allocating any new nodes; rearrange the pointers in the `next` fields. There is *not* a dummy node at the head of the list; the first node contains the first actual element.

Your name _____ login c60b–_____

Question 6 (6 points): Find, *explain*, and *fix* the bugs in the following programs. Each program has exactly one bug. (You may need to change more than one line to fix the bug, however.)

(a) In the graph programming project, you want to mark all the vertices as unvisited, so you say

```
    for (i = 1; i <= max_vertex; i++)
        unvisit(vertices[i]);
```

Here is the procedure `unvisit`:

```
void unvisit(struct vertex v)  {
    v.visited = 0;
}
```

(b) This procedure takes a positive integer and a list as arguments. It adds the integer to the *end* of the list. Assume that the list is *not* empty initially.

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

void append(int n, struct node *list) {
    struct node newnode;

    while (list->next != NIL)
        list = list->next;
    newnode.value = n;
    newnode.next = NIL;
    list->next = &newnode;
}
```

Question 7 continues on the next page.

Question 7 continued.

(c) This procedure reads a line from the keyboard, converts all letters to lower case, and puts the result into a preallocated buffer whose address is given as argument.

```
void user_input(char *buffer) {
    char response[100];
    char *cp;

    fgets(response, 100, stdin);
    for (cp = response; *cp; cp++) {
        *buffer++ = tolower(*cp);
    }
}
```