CS 60B Final — December 7, 1993

Your name _____

login c60b–_____

Discussion section number _____

TA's name _____

This exam is worth 30 points, or 30% of your total course grade. The exam contains seven questions.

This booklet contains ten 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	/5
6	/5
7	/6
total	/30

Question 1 (3 points):

(a) Convert the decimal number 726 to octal. Show your work.

(b) In some display systems a color is represented by a red value between 0 and 31, a green value between 0 and 31, and a blue value between 0 and 31. How many bytes are needed to represent the color at a single point?

(c) In the following MIPS program excerpt:

```
loop:  lb   $8, 0($9)
       addi $9, $9, 1
       bne  $8, $0, loop
```

translate the last instruction (the `bne`) to hexadecimal machine language.

Your name _____ login c60b–_____

Question 2 (3 points):

Explain *briefly* (no more than two sentences) a reason in support of each of the following claims:

(a) Some kernel memory references must be uncached.

(b) A large page size is good.

(c) A small page size is good.

Question 3 (4 points):

Translate the following fragment of C code into MIPS assembler:

```
void ReverseInPlace(char *s) {
    char *t = s;
    char ch;

    while (*t) t++;
    while (t > s) {
        ch = *t;
        *t-- = *s;
        *s++ = ch;
    }
}
```

Follow the MIPS register use conventions. Arguments are passed in registers.

Your name _____ login c60b–_____

Question 4 (4 points):

Given the following definitions:

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

typedef struct node *List;

#define NIL ((List)0)
```

translate the following fragment of C code into MIPS assembler. It separates a list into two lists, alternating the elements between them.

```
void split(List p, List *a, List *b) {
    if (p == NIL) {
        *a = NIL;
        *b = NIL;
        return;
    }
    *a = p;
    split(p->next, b, &p->next);
}
```

Follow the MIPS register use conventions. Arguments are passed in registers.

Question 5 (5 points):

Write a C function `IsSubstring` that takes two character strings as arguments, and returns true (nonzero) if and only if the first one appears as a consecutive substring of the second:

`substring("abc", "123abc456")` should return a nonzero integer
`substring("abc", "abc")` should return a nonzero integer
`substring("abc", "...a...b...c")` should return `0`
`substring("abc", "ab123abc")` should return a nonzero integer

Note: The last two examples above point out common bugs that you should try to avoid. The first substring must appear as *consecutive* characters in the second, but you shouldn't give up too soon if a partial substring appears.

Your name _____ login c60b–_____

Question 6 (5 points):

For your convenience here are the data structure definitions from the last C project:

```
struct place {
   char name[32];             /* name */
   struct place *nbr[4];      /* neighbours */
   struct thing *obj;         /* list of things in the place
                                 not owned by anyone */
   struct person *prs;        /* list of persons in the place */
   struct place *next;
};

struct person {
   char name[32];             /* name */
   struct place *loc;         /* current location */
   struct thing *pssn;        /* list of things possessed by the person */
   struct person *next;       /* next person in this same place */
};

struct thing {
   char name[32];             /* name */
   struct person *ownr;       /* owner if any */
   struct thing *next;        /* next unowned thing in this same place */
};
```

In one famous adventure game, called Wumpus, the player can smell the Wumpus (a sort of person) if it's one step away from the place where the player is. (If the Wumpus is in the same place as the player, then the Wumpus eats the player, and the game is over.)

We will add a new data structure to allow us to create lists of people, not all of whom are in the same place:

```
struct PersonListNode {
   struct person *who;
   struct PersonListNode *next;
};
```

Write a C function `neighbors` that takes a `struct person` as its argument, and returns a list of all the people who are one place away from that person in any direction.

Write your code on the following page!

Question 6 code here:

Your name _____ login c60b–_____

Question 7 (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) A procedure to test whether a number is in a list:

```
struct node {
    int value;
    struct node *next;
};
    typedef struct node *List;
#define NIL (List)0

int IsMember(int number, List p) {
    while (p != NIL) {
        if (p->value == number) return 1;
        p++;
    }
    return 0;
}
```

(b) A procedure to check whether a non-empty string contains all the same character, like "aaaa":

```
int AllSame(char *s) {
    int ch = *s++;

    while (*s++ = ch) ;
    if (*--s != '\0') return 0;
    return 1;
}
```

Question 7 continues on the next page.

Question 7 continued.

(c) A procedure to delete a number from a list (not freeing memory):

```
struct node {
    int value;
    struct node *next;
};
    typedef struct node *List;
#define NIL (List)0

void delete(int number, List p) {
    List old = p;

    while ((p = p->next) != NIL) {
        if (p->value == number) {
            old->next = p->next;
            return;
        }
        old = p;
    }
}
```