Midterm #3
27 April, 1994

This is an open books, open notes exam to be turned in by 10:50 AM.

Name: __________________________________________

Problem 1. (20 points)

Write a C procedure that takes a pointer to an integer as an argument and multiplies the integer by two. Call the procedure DoubleIt and start with the following outline:

```c
DoubleIt(int *pI)
{
    /* You add stuff in here */
}
```

Suppose your program contains an array A of 15 integers. Write a loop that will double each element of A by calling DoubleIt while passing, one at a time, each element of A. Start with the following declarations:

```c
int A[15], I;
```

Problem 2. (20 points)

Consider each of the following four loops

```c
/* Loop A */
for (i=0; i<n; ++i)
    for (j=i; j<n; ++j)
        s += A[i][j];

/* Loop B */
for (i=0; i<10; ++i)
    for (j=1; j<n; ++j)
        s += A[i][j];

/* Loop C */
for (i=0; i<1000; ++i)
    for (j=1; j<i; ++j)
        s += A[i][j];

/* Loop D */
for (i=n; i>0; --i)
    for (j=1; j<n; j *= 2)
        s += A[i][j];
```

In big-O notation, the running times of these algorithms are $O(1)$, $O(n)$, $O(n \log n)$, and $O(n^2)$. Match each algorithm with its running time.
Problem 3. (20 points)

Consider the following binary search algorithm for seeking an item \( k \) within an array \( A \).

\[
\text{low} = 0; \\
\text{high} = \text{MAXSIZE} - 1; \\
\text{while} \ (\text{low} <= \text{high}) \ {\{} \\
\quad \text{mid} = (\text{low}+\text{high}) / 2; \\
\quad \text{if} \ (A[\text{mid}] < k) \\
\quad \quad \text{low} = \text{mid}+1; \\
\quad \text{else} \\
\quad \quad \text{high} = \text{mid}; \\
\ {\}}
\]

If \( A \) is of size seven and contains the elements

\[14 \ 20 \ 23 \ 27 \ 33 \ 40 \ 48\]

and \( k \) is 21, what are the values of \( \text{low} \) and \( \text{high} \) at the beginning of each iteration of the binary search loop?

The two remaining problems are concerned with doubly linked lists. Use the following C data structure definition in your answers.

```c
typedef struct dl_node_tag {
    float name ;
    struct dl_node *next ;
    struct dl_node *prev ;
} DL_Node_type ;
```

PS: This should look familiar.

Problem 4. (20 points)

Starting with the variable \( pL \) pointing to the leftmost node of the doubly linked list on the right, show the result of executing the following five C statements in order.

\[
pL->\text{prev} = pL->\text{next} ; \quad /* 1 */ \\
pL->\text{next}->\text{next} = pL->\text{next} ; \quad /* 2 */ \\
pL->\text{next} = pL->\text{prev}->\text{next} ; \quad /* 3 */ \\
pL->\text{name} = 1492.0 ; \quad /* 4 */ \\
pL = pL->\text{prev} ; \quad /* 5 */
\]

You may find it useful to use the numbers in the comments to document the changes.
Problem 5. (20 points)

Again, suppose you have two pointers, pLeft and pRight, declared as follows:

```c
DL_Node_type *pLeft, *pRight;
```

that point to the left and right ends of a doubly linked list.

This time write a few lines of C to perform a "circular left shift" on the list. That is move the present left end of the list to follow the present right end of the list. Set pLeft and pRight to point to the new ends of the list.