Name:
Login Name:
Preceptor Name:
Precept Number:

Computer Science 126 — First Midterm Exam

10/20/1999 **7pm - 8:30pm**

This exam has 10 questions. The weight of each question is printed in the table below and next to each question. Do all of your work on these pages (use the back for scratch space), giving the answer in the space provided. Put your name on each page (now). Sign the Honor Code pledge.

"I pledge my honor that I have not violated the Honor Code during this examination."

Question	Worth	Earned
1	5	
2	7	
3	7	
4	7	
5	6	
6	6	
7	5	
8	6	
9	4	
10	7	
Total	60	

Name:

1. Debugging C [5]

The following program finds the value that occurs most often in an integer sequence of values between 0 and 99. Although this program will compile, there are some bugs in the code that will prevent it from calculating the correct answer. Find the errors and make the necessary corrections.

```
#include <stdio.h>
void
main()
{
   int num, i, maxi, a[99];

   while (scanf("%2d", num) != EOF)
      a[num]++;

   for (i = 0; i <= 99; i++);
      if (a[i] > a[maxi]) maxi = i;

   printf("%d\n", maxi);
}
```

Error 1:

Error 2:

Error 3:

Error 4:

Error 5:

Name:

2. Unix Standard I/O Redirection [7]

The following program is compiled to obtain an “a.out” file.

```
#include <stdio.h>
void
main ()
{
    int n;
    scanf ("%d", &n);
    printf ("%d is output.\n", n+1);
}
```

Assume the following:

a) the file "input" has a single number "58" in it; and
b) if keyboard input is needed, you will type "58".

Answer what the following commands do by filling out the table. If the command results in an error, write "yes" in the "error?" column and leave everything else blank.

	command	error ?	keyboad input ?	terminal output ?	output file ?	output text
0	a.out		yes	yes		59 is output.
1	input < a.out					
2	input | a.out					
3	a.out < input					
4	a.out > output					
5	a.out > a.out					
6	a.out | a.out					
7	a.out < input | a.out					

Name:

3. Structures [7]

Given the following definition:

```
typedef struct {float a; float b;} Interval;
```

Define a type for rectangles (`Rect`) that are parallel to the axes in a Cartesian coordinate system. Use the type `Interval` defined above.

Write a function that returns 1 if a point falls within a rectangle, 0 otherwise. Use the function prototype below and define your own `Point` type.

```
int inrect(Point, Rect);
```

Name:

4. Pointers and Arrays [7]

What does the following program print?

```
#define N 6

main() {
    int i;
    int a[N];
    int *p, *q;

    p = &a[N-1];
    q = p - (N-1);

    for (i = 0; i < N; i++) {
        *(p-i) = i;
        *(q+i) = i;
    }

    for (i = 0; i < N; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}
```

Name:

5. Pointers and Linked Lists [6]

Suppose that a linked list is made up of nodes of type

```
typedef struct node *Link;
struct Node {int key; Link next; };
```

that you are given a pointer **`list`** of type Link, which points to the first node of the list; that the last node has **`NULL`** as its link; and that there are at least two nodes on the list. Write a code fragment to add a new node just after the second node of the list.

Name:

6. Linked Lists and Recursion [6]

```
typedef struct node *Link;
struct Node {int key; Link next; };

int
f(Link list)
{
    int a,b;
    if (list == NULL)
        return 0;
    a = list->key;
    b = f(list->next);
    if (a > b)
        return a;
    else
        return b;
}
```

Suppose the above function is given a list whose content is `(9 8 1 12 8 2 5 7)`. (The head of the list contains the key 9. The `next` link of the last element is `NULL`.) What does the above function return?

Suppose the above function is given a list whose content is `(-9 -8 -1 -12 -8 -2 -5 -7)`. What does the above function return?

Name:

7. ADT: Stacks and Queues [5]

Suppose we have an empty stack of integers and an empty queue of integers to begin with, what will the following code fragment print?

```
for (i = 0; i < 100; i++) stackPush(i);
for (i = 0; i < 100; i++) queuePut(stackPop());
for (i = 0; i < 100; i++) stackPush(queueGet());
for (i = 0; i < 100; i++) printf("%d ", stackPop());
```

Name:

8. Recursion [6]

Given the recursive solution of the "Traveling Salesman Problem" discussed in class:

```
visit(int N) {
    int i;
    if (N == 1) {
        checkLength();
        return;
    }
    for (i = 1; i <= N; i++) {
        swap(i, N);
        visit(N-1);
        swap(i, N);
    }
}
```

When executing `visit(4)`, how many times does it invoke `checkLength()`?

When executing `visit(4)`, how many times does it invoke `visit(2)`?

Name:

9. Searching Arrays [4]

You have learned binary search. Let us extend it to "ternary search". In binary search, we repeatedly divide the search range into two roughly equal halves and search only one of them. In a ternary search, we repeatedly divide the search range into three roughly equal parts and search only one of them. In the worst case, how many comparisons do we make to search an array of N elements?

A) $\mathrm{Log}_2(N)$

B) $\mathrm{Log}_3(N)$

C) $3*\mathrm{Log}_2(N)$

D) $2*\mathrm{Log}_3(N)$

E) $2*\mathrm{Log}_2(N)$

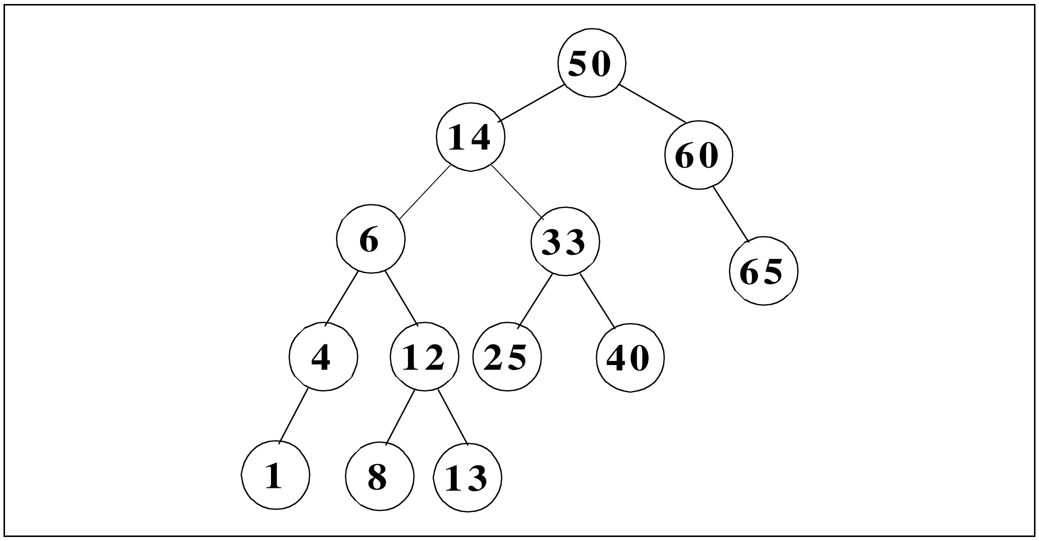
F) $3*\mathrm{Log}_3(N)$

Briefly justify your answer (with one or two sentences or a simple picture).

Name:

10. Trees [7]

You are given the following binary search tree:



We perform a preorder traversal using a stack. During the traversal, at the moment when the top of the stack is 8, what is the content of the entire stack? As a reminder, here is the traversal code:

```
void
preorder(Link h) {
    stackPush(h);
    while (!stackEmpty()) {
        h = stackPop();
        visit(h);
        if (h->r != NULL)
            stackPush(h->r);
        if (h->l != NULL)
            stackPush(h->l);
    }
}
```