Question #1:

Consider the following recursive function:

```
int f(int x)
{
     if (x <= 0)
          return 5;
     else
          return f(x - 1) + 3;
}
```

a) What final value will the function return if it is passed the argument 0?

b) What final value will the function return if it is passed the argument 1?

c) What final value will the function return if it is passed the argument 50?

Question #2:

Assume an array has been declared as follows:

```
int num[100];
```

Also assume that p1 and p2 are pointers that have been declared as follows:

```
int *p1;
int *p2;
```

a) Write a statement that will make p1 point to the first element of the array.

b) Write a statement that will make p2 point to the last element of the array.

c) After these two statements are executed, what will be the value of p2 – p1 (using pointer arithmetic)?

d) Assuming that you are dealing with a system such that each integer requires 32 bits (4 bytes), what will be the actual difference between the memory addresses stored in p1 and p2?

Question #3:

A programmer wants to write a program that first prompts the user to enter a number in the range of 5 to 20 (repeatedly asking until an integer in this range is entered), and then displays a square of asterisks, using the specified number as the number of rows and columns. (For example, if the user enters 10, the program should display 10 rows each containing 10 asterisks.)

The programmer writes the following code:

```
#include <stdio.h>

int main(void)
{
     int x;
     int y, z;

     do
     {
          printf("Please enter a number from 5 to 20: ");
          scanf("%d", x);
     } while ((x < 5) && (x > 20));

     for (y = 1; y <= x; y++);
     {
          for (z = 1; z <= y; z++)
               putchar('*');
     }

     return 0;
}
```

The above code has 5 errors in it! What are they?

Question #4:

You compile and run the following program:

```
#include <stdio.h>

int x = 100;

int do_stuff(int *, int *);

int main(void)
{
     int z = 200;
     int *p1 = &x;
     int *p2 = &z;

     x = do_stuff(p1, p2);

     printf("x=%d, z=%d, *p1=%d, *p2=%d\n", x, z, *p1, *p2);

     return 0;
}

int do_stuff(int *p1, int *p2)
{
     int z = 300;

     *p1 = 400;
     p2 = &z;
     *p2 = x + z;

     printf("x=%d, z=%d, *p1=%d, *p2=%d\n", x, z, *p1, *p2);

     return (*p1 + *p2);
}
```

What gets displayed to standard output?

Question #5: (15 points)

The following are the definitions concerning nodes in a linked list:

```
typedef struct node
{
     int x;
     struct node *next;
} NODE;

typedef NODE *PNODE;
```

Write a function called "compute_average" that accepts, as a parameter, a pointer "pList" which points to the head of a linked list. The function should compute and return the average (arithmetic mean) of all of the integers stored in the linked list. The function should not assume that the average is an integer. If the list is empty, the function should return 0.
