Condition Driven Loops

1. Consider the following program.

```
#include <stdio.h>
int main( void ) {
      double a,b;
      int ret_val;

      printf("enter two numbers separated by a space: ");
      ret_val = scanf("%lf %lf", &a, &b);

      printf("a: %.1f, b: %.1f, returned value: %d\n", a, b, ret_val);
}
```

What is the output when the program is run with the following keyboard inputs entered? (indicate garbage value with –)

a) `5 7`

a: 5.0, b: 7.0, returned value: 2

b) `5.4 7.25`

a: 5.4, b: 7.2, returned value: 2

c) `abc 5.7`

a: –, b: –, returned value: 0
the abc is not scanned successfully and so blocks scanf from scanning the next value

d) `abc efg`

a: –, b: –, returned value: 0

e) `3.8 82ef`

a: 3.8, b: 82.0, returned value: 2
but the "ef" will remain as input and the next call to scanf will read the "ef"

2. Design a function that repeatedly prompts a user to enter integers one at a time, until the user enters the value -1 to indicate that there are no more values. The function should count the number of odd positive values the user enters and return that count.

   Update your function so that if the user enters data that is not valid integer, the loop terminates and the function prints an error message and returns -1.

```
#include <stdio.h>

int count_odd_positive();

int main( void ) {
    int num_odd_pos = count_odd_positive();
    printf("%d\n", num_odd_pos);

    return 0;
}

/*
 * Purpose: collects numbers from standard input until -1 is entered,
 *   and counts the number of positive odd numbers entered.
 *  If invalid data is entered an error message is printed.
 * Returns: int - the count of positive odd numbers entered,
 *    or -1 if invalid data is entered.
 */
int count_odd_positive() {
    int scanned, next_val, count = 0;

    printf("enter an integer, -1 to stop\n");
    scanned = scanf("%d", &next_val);

    while(scanned == 1 && next_val != -1) {
        if (next_val>0 && next_val%2==1) {
            count++;
        }
        printf("enter a number 1-100, -1 to stop\n");
        scanned = scanf("%d", &next_val);
    }

    if (scanned != 1) {
        printf("error in data entry %d\n", count);
        count = -1;
    }

    return count;
}
```

3. Design a function called `stats`, that repeatedly prompts a user to enter a positive integers one at a time, until the user enters the value -1 to indicate that there are no more values. The function should take 2 integer pointers as arguments, one to hold the pointer to smallest value entered by the user and one to hold the pointer to the largest value entered by the user. If the first number entered by the user is -1, both the smallest and largest will be set to -1 and a message will be printed: *Error - no positive numbers entered*
The function should assume the user will only enter valid data (integers >= -1).

   HINT: you will need to keep track of the smallest and largest values seen so far as you handle each value entered by the user.

```
#include <stdio.h>

void stats(int* smallest, int* largest);

int main( void ) {
    int min, max;
    stats(&min, &max);
    printf("smallest: %d, largest: %d\n", min, max);

    return 0;
}


/*
 * Purpose: collects numbers from standard input until -1 is entered,
 *   and tracks the smallest and largest numbers entered.
 *  Prints an message if no positive values entered.
 *  The function assumes user is only entering valid integers >= -1.
 * Parameters: int* smallest - pointer to the smallest number entered
 *             int* largest - pointer to the largest number entered
 */
void stats(int* smallest, int* largest) {
    int next_val, min, max;

    printf("enter an integer, -1 to stop\n");
    scanf("%d", &next_val);

    min = next_val;
    max = next_val;

    while(next_val != -1) {
        if (next_val<min)
            min = next_val;
        else if (next_val>max)
            max = next_val;

        printf("enter an integer, -1 to stop\n");
        scanf("%d", &next_val);
    }

    if(min == -1)
        printf("Error - no positive numbers entered\n");

    *smallest = min;
    *largest  = max;
}
```

4. Design a function that takes two arguments representing a minimum value and a maximum value. The function should repeatedly asks the user to enter a number in the range minimum value to maximum value inclusive. The function should return the valid number entered. The function can assume the user will only enter integers.
5. Design a function that computes the average of a bunch of non-negative integers between (0 and 100 inclusive) entered by the user. The user enters the numbers, one at a time, and enters the value -1 to indicate that there are no more values to enter. The average is computed excluding the -1 and returned from the function. If no values between 0 and 100 were entered, the function should return 0.
   Your function can assume the user will only enter integers but…
   If the user enters a number not between 0 and 100 or not -1, they are repeatedly prompted until they enter a valid number. Any integers outside the range of 0 and 100 are not included in the average computation.
   TIP: think about how to use the function you just wrote.

```
#include <stdio.h>

int get_number(int min, int max);
double get_average();

int main( void ) {
    int num = get_number(-8, 32);
    printf("should be a number in range (-8, 32): %d\n", num);

    double avg = get_average();
    printf("should be average of numbers entered): %f\n", avg);

    return 0;
}

/*
 * Purpose: repeatedly prompts a user for a number between min and max inclusive.
 *  The function assumes user is only entering integers.
 * Parameters: int min - the minimum value to be entered,
 *             int max - the maximum value to be entered, min<=max
 * Returns: int - the valid number entered
 */
int get_number(int min, int max) {
    int num;

    printf("enter an integer between %d and %d inclusive\n", min, max);

    //Notice: not storing value returned from scanf, function assumes only integer input
    scanf("%d", &num);

    while(num < min || num > max) {
        printf("no, an integer between %d and %d inclusive\n", min, max);
        scanf("%d", &num);
    }

    return num;
}

/*
 * Purpose: collects numbers from standard input until -1 is entered,
 *   and calculates the average of the positive numbers entered.
 *   The function assumes user is only entering integers.
 * Returns: double - the average, or 0 if no positive numbers entered.
 */
double get_average() {
    int num, count=0, sum=0;
    double avg;

    num = get_number(-1,100);

    while(num!=-1) {
        count++;
        sum += num;
        num = get_number(-1,100);
    }

    if (count==0) {
        avg = 0;
    } else {
        avg = sum/(double)count;
    }
    return avg;
}
```