Condition-driven Loops

1. Consider the functions below. Update the documentation, function and variable names to reflect their behavior.

```
/*
 * Purpose: counts the number of vowels entered by the user
 * Parameters: none
 * Returns: int, the count
 */
int count_vowels( ) {
    int s = 0;
    char ch;

    ch = getchar();
    while( ch != '#' ) {
        if (is_vowel(ch) == 1) {
            s++;
        }
        ch = getchar();
    }
    return s;
}

/*
 * Purpose: determines whether c is a vowel
 * Parameters: char c
 * Returns: int, 1 if c is a vowel, 0 otherwise
 */
int is_vowel(char c) {
    int ret_val;

    if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u' ||
       c=='A' || c=='E' || c=='I' || c=='O' || c=='U')   {
        ret_val = 1;
    } else {
        ret_val = 0;
    }

    return ret_val;
}
```

2. Design a function that will compute and return the number of characters entered by the user that are not a space (ie. a newline, tab, space, etc). The user will enter an arbitrary number of characters and finally the value # to indicate that there are no more values to enter.
Recall the functions included in `ctype.h`: `int getchar()` and `int isspace(int c)`

```
/*
 * Purpose: counts the number of non-space characters entered by the user
 * Parameters:  non-space
 * Returns: int - the count of non-space characters
 */
int count_characters( ) {
    char ch;
    int count = 0;

    ch = getchar();
    while( ch != '#' ) {
        if (!isspace(ch)) {
            count++;
        }
        ch = getchar();
    }
    return count;
}
```

3. Design a function that will compute and return the number of characters in the first word entered by the user. When any kind of space is encountered (ie. a newline, tab, space, etc), this indicates the first word is complete. At this point the function should terminate and return the number of characters in that word. The user will enter an arbitrary number of characters and finally the value # to indicate that there are no more values to enter. For simplicity sake, you can count all characters including numbers and characters that are not a space as part of the word. You must take into account the case where no spaces are entered before the #, in this case, the function should return the number of characters before the #.
// solution: counts a word after seeing the last letter (sees a space), spaces not considered a word

```
/*
 * Purpose: counts the number of characters entered by the user
 *  before a space character or # character
 * Parameters:  none
 * Returns: int - the count of characters
 */
int count_characters_in_word( ) {
    char ch;
    int letter_count = 0;

    ch = getchar();
    while (!isspace(ch) && ch != '#') {
        letter_count++;
        ch = getchar();
    }
    return letter_count;
}
```

4. Design a function that takes a character, `ch` and will compute and return the number of characters entered by the user that match `ch`. The user will be prompted to enter an arbitrary number of characters where they enter the value # to indicate that there are no more values to enter.

```
/*
 * Purpose: count the number of characters entered by the user that match ch
 * user enters '#' to indicate the end of input
 * Parameters:  char ch
 * Returns: int - the count of matching characters
 */
int count_matching(char ch) {
    char next;
    int count = 0;

    next = getchar();
    while( next != '#' ) {
        printf("%c\n", ch);
        if (next == ch) {
            count++;
        }
        next = getchar();
    }
    return count;
}
```

5. Design a function that will compute and print the percentage of characters entered by the user that are uppercase and the percentage of characters that are lowercase. The user will enter an arbitrary number of characters and finally the value # to indicate that there are no more values to enter.
Print the percentages with 1 significant figure. If no text was entered, print "no text entered".
For example, if the user enters the following 5 characters: `b 6a$BCf#`
The function should print: 25.0% uppercase and 37.5% lowercase since there are 2 uppercase letters, 3 lowercase letters and 3 other non-letter characters (space, 6 and $).
Recall the functions included in `ctype.h`: `int isupper(int c)` and `int islower(int c)`

```
/*
 * Purpose: count the number of characters entered by the user that match ch
 *          The user enters '#' to indicate the end of input
 * Parameters:  none
 * Returns: none
 */
void print_percent_upper_lower() {
    char next;
    double percent_upper;
    double percent_lower;
    int count_upper = 0;
    int count_lower = 0;
    int count = 0;

    next = getchar();
    while( next != '#' ) {
        if (isupper(next)) {
            count_upper++;
        } else if (islower(next)) {
            count_lower++;
        }
        count++;
        next = getchar();
    }
    if (count != 0) {
        percent_upper = (double)count_upper/count * 100;
        percent_lower = (double)count_lower/count * 100;

        printf("%.1f%% uppercase, %.1f%% lowercase\n", percent_upper, percent_lower);
    } else {
        printf("no text entered\n");
    }
}
```