```c
/* lab1.c
 * Richard Coffey
 * rrcoffe
 * ECE 2220, Fall 2016
 * MP1
 *
 * NOTE:  You must update all of the following comments!
 *
 * Purpose: This program is intended to detect the start of incoming radio transmis
sions. The program takes input from the user, in the form of a correlation threshol
d value, a minimum correlation threshold, and sample values. If the program detects
 a value from the given samples greater thn or equal to the minimum correlation val
ue, and a number of times greater than or equal to the correlation threshold, it ou
tputs an analysis of the sample.
 *
 * Assumptions: We assume that the incoming transmission will be numbers only, not
letter characters. We also assume the input will be uniform, and that either the in
put file or manual entry will be appropriate for the input method.
 *
 * Bugs: This program will only accept up to 32bit numbers, any sample value greate
r than 2,147,482,647 will overflow the memory value.
 *
 * See the ECE programming guide for requirements
 *
 * Are you unhappy with the way this code is formatted?  You can easily
 * reformat (and automatically indent) your code using the astyle
 * command.  If it is not installed use the Ubuntu Software Center to
 * install astyle.  Then in a terminal on the command line do
 *     astyle --style=kr lab1.c
 *
 * See "man astyle" for different styles.  Replace "kr" with one of
 * ansi, java, gnu, linux to see different options.  Or, set up your
 * own style.
 *
 * To create a nicely formated PDF file for printing install the enscript
 * command.  To create a PDF for "file.c" in landscape with 2 columns do:
 *     enscript file.c -G2rE -o - | ps2pdf - file.pdf
 */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

//notice there is no semi-colon at the end of a #define
#define MAXLINE 100
#define MAXSAMPLES 500
#define STOPCOUNT 3
#define MINTHRESH 3

int main()
{
    char line[MAXLINE];
    int corr_thresh = 0;
    int min_corr_val = 0;
    int samples[MAXSAMPLES];
    int i = 0;
    int j = 0;
    int a = 0;
    int b = 0;
    int sortNum;
    int arrayiter = 0;
    int unsortedArray[MAXSAMPLES];
    int unsortedCount;

//Configuration Stage

    printf("What is the correlation threshold? ");
    fgets(line, MAXLINE, stdin);
    sscanf(line, "%d", &corr_thresh);
    if (corr_thresh == -1) {
        exit(-1);
    }
    printf("\nThe correlation threshold is equal to: %d\n", corr_thresh);

    // While loop to ask for correlation threshold until a good value is written
    while (corr_thresh < MINTHRESH) {
        printf("That is not a valid correlation threshold!\n");
        printf("Please enter a valid correlation threshold!\n");
        fgets(line, MAXLINE, stdin);
        sscanf(line, "%d", &corr_thresh);
        if (corr_thresh == -1) {
                exit(-1);
        }
        printf("\nThe correlation threshold is equal to: %d\n", corr_thresh);
    }

    printf("What is the minimum correlation value? ");
    fgets(line, MAXLINE, stdin);
    sscanf(line, "%d", &min_corr_val);
    if (min_corr_val == -1) {
        exit(-1);
    }
    printf("\nThe minimum correlation value is equal to: %d\n", min_corr_val);

    // While loop to ask for minimum correlation value until a good value is writte
n
    while (min_corr_val < 1) {
        printf("That is not a valid minimum correlation value!\n");
        printf("Please enter a valid minimum correlation value!\n");
        fgets(line, MAXLINE, stdin);
        sscanf(line, "%d", &min_corr_val);
        if (min_corr_val == -1) {
                exit(-1);
        }
        printf("\nThe minimum correlation value is equal to: %d\n", min_corr_val);
    }

//End Configuration Stage

    // Next collect the samples.
while (samples[0] != -1) {

    printf("\nPlease enter the sample values: ");

  //Stating variables within while loop so the variables get reset each time the lo
op runs
    int pos = 0;
    int val = 0;
    int candcount = 0;
    int count = 0;
    int samplesize = 0;
    int zerocount = 0;
    i = 0;
    arrayiter = 0;

    while (zerocount < STOPCOUNT && samplesize < MAXSAMPLES) {

        //Reading in the samples
        fgets(line, MAXLINE, stdin);
        sscanf(line, "%d", &samples[i]);
        if (samples[0] == -1) {
                printf("Exiting program, goodbye\n");
                exit(-1);
        }

        //Stop Count detection
        if (samples[i] == 0) {
                zerocount++;
        }

        if (samples[i] != 0) {
                zerocount = 0;
        }
        //End Stop Count Detection
        i++;
        samplesize++;

        }
        //Copying the array
        for(unsortedCount = 0; unsortedCount < i; unsortedCount++) {
                unsortedArray[unsortedCount] = samples[unsortedCount];
        }

        //Bubble Sort Algorithm
        for(a = 0; a < i; a++) {
                for (b = 0; b < i-1; b++) {
                        if (samples[b] > samples[b+1]) {
                                sortNum = samples[b+1];
                                samples[b+1] = samples[b];
                                samples[b] = sortNum;
                        }
                }
        }

        /*Determines:
        1. If the sample value is greater or equal to the minimum correlation value
        2. If the sample value matches the next sample value, it increments a count
 value
        3. If the count value is greater than the correlation threshold,the specifi
c value is assigned to 'var'
        4. If sample value does not match the next sample value, the count incremen
ter is reset.
        5. candcount preserves the count value in case the count incrementer is era
sed
*/
        count = 1;
        while (arrayiter < i) {
                if (samples[arrayiter] >= min_corr_val) {
                        if (samples[arrayiter] == samples[arrayiter+1]) {
                                count++;
                                if (count >= corr_thresh) {
                                        val = samples[arrayiter];
                                        candcount = count;
                                }
                        }
                        else {
                                count = 1;
                        }
                }
                arrayiter++;
        }

        //Finds first appearance of candidate sample value
        j = 0;
        while (pos == 0) {
                if (unsortedArray[j] == val) {
                        pos = j+1;
                }
                j++;
        }

    // Use one of the following prints
    // you must use exactly these prints and you cannot change the text
        if (val != 0) {
                printf("Waveform detected at position %d with value %d and appears
%d times\n", pos, val, candcount);
        }
        if (val == 0) {
                printf("No waveform detected\n");
        }


        }

        printf("Goodbye\n");
        exit(0);
}
```