```c
/*****************************************************************************
*                            countPalindrome
*****************************************************************************
Command format:

            contaPalindromi  <input.txt>  <output.txt>  <row>

Receives sentences from <input.txt> and writes the number of palindrome words in
the sentence indicated by <row>, and prints the sentence itself.

It checks the correctness of the command format (number of arguments passed and
value of <row>).
*/

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

# define MAX 1000

typedef unsigned short int U_S_INT;

unsigned int palCount(char []);
unsigned int FtoM(FILE *, char [][MAX]);
U_S_INT isPalindrome_1(char []);
U_S_INT isPalindrome(char[], int, int);



int main (int argc, char * argv[]) {
    if (argc != 4){
        printf("\n Incorrect command!\n");
        return 0;
    }

    int rows;
    char m[MAX][MAX];
    U_S_INT sentence;
    FILE *fin, *fout;

    fin  = fopen(*(argv + 1), "r");
    fout = fopen(*(argv + 2), "w");

    sentence = atoi(*(argv + 3));
    rows = FtoM(fin , m);

    if (sentence == 0) {
        printf("\nInvalid Argument (n. of Sentence)\n");
        return 0;
    }
    else if (sentence > rows) {
        printf("\nNot enough sentences in %s\n", *(argv + 1));
        return 0;
    }

    sentence--;

```

```c

    fprintf (fout, "%s\n\nSentence n.: %d\n\n", m[sentence], (sentence + 1));
    fprintf (fout, "N. of Palindrome words: %d\n\n", palCount(m[sentence]) );
    fprintf (fout, "END.\n");

    printf("\nResults in %s\n", *(argv + 2));

    fclose(fin);
    fclose(fout);
}


/*****************************************************************************
*                                palCount
*****************************************************************************

Reads 'r' and writes words into 'w'; then checks that 'w' is palindrome
with the function 'isPalindrome' and if it's the case, 'palindromes' is
incremented. The variable 'j' is used to detect when we're into a word.

NB: the function assumes that characters '-' and '_' can only be part of a word
and that '.' can be part of a word or at the end of a sentence.

NB: The presence of '\n' is not expected.
*/


unsigned int palCount (char r[]) {
    U_S_INT j = 0;
    int i = 0;
    int palindromes = 0;
    char w[MAX];

    while(r[i] != '\0') {
        if (j == 0 && (ispunct(r[i]) || isspace(r[i])))
            continue;
        else if ( j != 0
                  && (isspace(r[i]) || (ispunct(r[i])
                  && (r[i] != '-') && (r[i] != '_')
                  && (r[i] != '.') ))) {
            w[j] = '\0';
            if (isPalindrome(w, 0, j - 1))
                palindromes++;
            j = 0;
        }
        else if (j != 0 && (r[i] == '.' && r[i + 1] == '\0')) {
            w[j] = '\0';
            if (isPalindrome(w, 0, j - 1))
                palindromes++;
            j = 0;
        }
        else
            w[j++] = r[i];
        i++;
    }
    return palindromes;
}

```

```c

/*****************************************************************************
*                              isPalindrome
*****************************************************************************/

/* Two versions, one recursive, one iterative.*/

U_S_INT isPalindrome(char s[], int i, int j) {
    if (s[i] == s[j] && i < j)
        isPalindrome(s, i + 1, j - 1);
    else
        return 0;
    return 1;
}


U_S_INT isPalindrome_1(char s[]) {
    int i = 0, len, range;

    if ((len = strlen(s)) <= 2)
        return 0;
    else if (len % 2 != 0)
        range = (len - 1) / 2;
    else
        range =  len / 2;

    len--;

    while (s[i] == s[len - i] && i < range)
        i++;

    if (i == range)
        return 1;
    else
        return 0;
}
```

```c
/*****************************************************************************
*                                FtoM
*****************************************************************************/

unsigned int FtoM (FILE *f, char m[][MAX]) {
    int c, i=0, j=0;

    while ((c = getc(f)) != EOF && j < MAX) {
        /* Ignore initial spaces */
        if (isspace(c) && j == 0)
            continue;
        else if (c != '.' && c != '\n')
            m[i][j++] = c;
        else if (c == '.') {
            m[i][j++] = c;

            if ( isspace( (c = getc(f)) ) ) {
                m[i++][j] = '\0';
                j = 0;
            }
            else if (c == EOF) {
                m[i][j] = '\0';
                break;
            }
            else
                m[i][j++] = c;
        }
        else
            m[i][j++] = ' ';
    }
    return i + 1;
}

```