Write a page replacement algorithm in C according to the following conditions:

Check for 3 positions in the reference string, both in forward and backward direction, the page request which is recurring the least number of times has to be replaced from the memory.In case of a clash, select the numerically lesser page number for replacement. Consider the memory to be initially empty with size of 3 frames. Calculate the number of page faults:

Filename: pralgo.c

Example:

Before Replacement

1 0 0 1 2 3 2 2 0 ('2' is occurring three times, '0' is occurring two times and '1' is one time)

0
1
2

After Replacement:

1 0 0 1 2 3 2 2 0

0
3
2

```
Sample Input:
Enter the size of reference string:
9
Enter the reference string
1
0
0
1
2
3
2
2
0

Output:
4
```

```
Code:
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

// total number of frames
int number_of_frames = 3;

// total size fo reference sting
int size_of_ref;

// hold the current no of frame.
int current_size;


void schedule(int[]);
int getindex(int[], int[], int);



int main()
{
    //get the size of reference sring.
    scanf("%d", &size_of_ref);


    int reference_string[size_of_ref];

    for(int i=0; i< size_of_ref; i++)
    {
```

```
        scanf("%d", &reference_string[i]);
    }

    // number_of_frames = 3;
    schedule(reference_string);
    return 0;

}


void schedule(int reference_string[])
{
    int page_fault_count = 0;
    int frames[number_of_frames];
    int ref, rep_ind, success;

    for(int i=0; i<number_of_frames; i++)
    {
        frames[i] = -1;
    }

    for(int i=0; i<size_of_ref; i++)
    {
        ref = reference_string[i];
        success = 0;

        for(int i = 0; i< current_size; i++)
        {
            if(frames[i] == ref)
            {
                success = 1;
            }
        }

        if(success)
        {
            continue;
        }

        page_fault_count++;

        if(current_size < number_of_frames)
        {
            frames[current_size] = ref;
            current_size++;
            continue;
        }

        rep_ind = getindex(reference_string, frames, i);
        frames[rep_ind] = ref;
    }

    printf("%d\n", page_fault_count);

}


int getindex(int reference_string[], int frames[], int curr_ind)
{
    int minimum_occurences = -1;
```

```
    int repl_index = -1;
    int occurences;
    int j;

    for(int i=0; i<number_of_frames; i++)
    {
        occurences = 0;
        j = curr_ind-1;


        while( (j>=0) && (j>(curr_ind-4)))
        {
            if(frames[i] == reference_string[j])
            {
                occurences += 1;
            }
            j-=1;
        }
        j= curr_ind + 1;
        while((j<(curr_ind + 4)) && (j<(size_of_ref)))
        {
            if(frames[i] == reference_string[j])
            {
                occurences++;
            }
            j++;
        }

        if(occurences <= minimum_occurences)
        {
            if(repl_index == -1)
            {
                minimum_occurences = occurences;
                repl_index = i;
            }
            else if(repl_index != -1 && frames[i] < frames[repl_index])
            {
                minimum_occurences=occurences;
                repl_index =i;
            }
        }
    }

    return repl_index;
}
```

Write a shell program for the following operation:
Get a sentence from the user (all words in lower-case) such that it ends with “done”. Count the number of words (n) excluding done and print the count followed by central word (if count is even then n/2-1th word and if it odd n/2th). n should be minimum 4.
Sample:

Input:
Hello
my
name
is
something
done

Output:
5
name

Code:

```
#! /bin/bash

#array of string to store the wordws of sentence.
sentence=()

while [ true ];
do

    # get input from user.
    read word

    # If word is done, break loop.
    if [ $word = "done" ]
    then
        break;
    fi

    # ammend word to sentence array.
    sentence+=($word)

done


#Counting the total number of words in the sentence array.
word_count=${#sentence[@]}

#display the number of words.
```

```
echo $word_count


# checking whether word count is odd  or even.

remainder=`expr $word_count%2`
if [ $remainder -eq 0]
then
    center=$(($word_count / 2))
    center=$(($center - 1))
else
    center=$(($word_count / 2))
fi

# extracting the middle one from the sentence.
middle_string=${sentence[$center]}

# Displaying the middle word in small.
echo ${middle_string,,}
```