Problem 1:
Write a C Program to determine whether a word/a line/a number is palindrome or not?

Before we start to write the program, we need to know the points discussed below.

a) A **palindrome** is a word, or a sentence or a number that is the same whether we read it backward or forward.
Example –
Word: civic, madam, level, refer, radar, malayalam **etc.**
Number: 1881, 121 **etc**
Sentence: Noel sees Leon

b) **Comma operator:**
The **comma operator** (,) is used to link the related expressions together. It has the lowest precedence of any other C operators. A list of expressions separated by comma is evaluated left to right and the value of right-most expression is the value of the combined expression.
Example:

```
c = (a = 5, b = 10, a + b);
```

5 is assigned to a, then 10 is assigned to b and finally 15 (5 + 10) is assigned to c.
Note here **parentheses are necessary**. As comma operator has the lowest precedence of all operators.
Comma operator most often finds use in **for loops**.

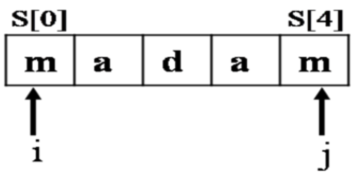
Example:

```
for(i =1 , j = 5; i <= j; i++, j--)
        printf("\ni = %d, j = %d", i, j);
```

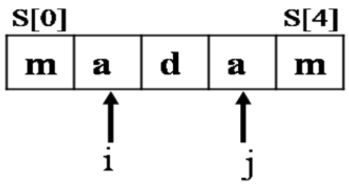
Review Question 1:
Write the output of the above code segment?

c) **Methodology that we will follow to check given string is palindrome or NOT is shown below –**

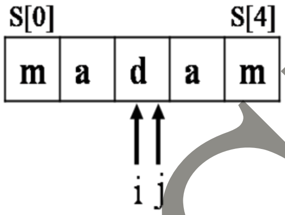
Let string entered be **madam**
Length of the string , len = **5**
We will use index **i** and index **j** for comparison of letters.



Initially, **i = 0**, **j = len - 1 = 4** ,
Test condition **S[i] != S[j]**
FALSE, go for **next iteration**

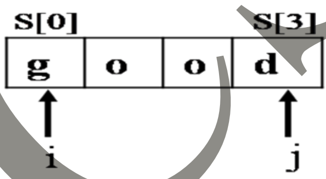


i = 0 + 1 = 1, **j = 4 – 1 = 3,**
Test condition **S[i] != S[j]**
FALSE, go for **next iteration**



i = 1 + 1 = 2, **j = 3 – 1 = 2**,
As NO need to Test **S[2]** with **S[2]**,
So **LOOP** must **terminate** now and **given string** is a **palindrome.** Hence, condition to terminate loop will be **i < len / 2**. [**integer division**]

If string entered is **good** , then len = **4**.



Initially, **i = 0**, **j = len - 1 = 3**,
Test condition **S[i] != S[j]**
TRUE, terminate loop and declare given string is **NOT** a **palindrome.**

d) If given string is **Madam** then also it should be treated as a palindrome. But when **M** will be compared with **m** then **M != m**. So in order to handle such situation, we will convert each letter either in **uppercase** or in **lowercase** before comparison.

Now we will write the program for problem 1.

--

Problem 1:
Write a C Program to determine whether a word/a line/a number is palindrome or not?

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main()
{
        char str[100];
        int i, j, len;
        printf("\nEnter the string : ");
        gets(str);
        len = strlen(str);
        for(i = 0, j = len – 1; i < len/2; i++ , j--)
        {
                if(toupper(str[i]) != toupper(str[j]))
                {
                        break;
                }
        }
        if(i == len/2)
                printf("\n%s is a Palindrome", str);
        else
                printf("\n%s is not a Palindrome", str);
        return(0);
}
```

The above program is checked using following test cases.

Test 1: Enter the string : **Madam**
Madam is a Palindrome

Test 2: Enter the string : **good**
good is not a Palindrome

Test 3: Enter the string : **1881**
1881 is a Palindrome

Test 4: Enter the string : **ABLE WAS I ERE I SAW ELBA**
ABLE WAS I ERE I SAW ELBA is a Palindrome

Test 5: Enter the string : **Noel sees Leon**
Noel sees Leon is a Palindrome

<u>Review Question 2</u>:

Write a C Program without using string & character handling functions to determine whether a word/a line/a number is palindrome or not?

Hint:

We **cannot** use strlen( ) and toupper( ) functions.

1) To determine length of the string, use below steps.
    - initialize len = 0.
    - Then use the following code segment

```
while(str[len] != '\0')
        len++;
```

2) Use mytoupper() to convert a character to uppercase.
    - Function declaration is as follows –

```
char mytoupper(char);
```

    - Function definition may be as follows

```
char mytoupper(char c)
{
        if(c>=97)
                return(c-32);
}
```

Problem 2:
WACP to count the vowels, consonants, digits, white space and special characters present in a line.

```
#include <stdio.h>
#include <ctype.h>
int main()
{
        char c,str[100];
        int i, dc, vc, cc, sc, wc;
        dc=vc=cc=sc=wc=0;
        printf("\nEnter the string : ");
        gets(str);
        for(i=0; str[i] != '\0'; i++)
        {
                c=toupper(str[i]);
                if(c>='0' && c<='9')
                        dc++;
                else if(c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U')
                        vc++;
                else if(c > 'A' && c < 'Z')
                        cc++;
                else if(ispunct(c))
                        sc++;
                else
                        wc++;
        }
        printf("\n         Digit Count = %d",dc);
        printf("\n         Vowel Count = %d",vc);
        printf("\n     Consonant Count = %d",cc);
        printf("\nSpecial Character Count = %d",sc);
        printf("\n    White space Count = %d",wc);
        return(0);
}
```

Problem 3:
WACP that takes the **full name** of a person as **input** and **prints** the first letters of first and middle name (if any) and the **title** as it is. For example the program should print **R.N.Tagore** for an input **Rabindra Nath Tagore.**

```
#include <stdio.h>
#include <string.h>
int main()
{

    /* Declaring three character arrays, first one to store input given, Second one to store
   each word of the input and third one to Store modified string */

        char a[50],word[20],b[50];
        int i,j=0,k=0;

  //Getting the Full Name from user
        printf("\n Enter Full Name of a person : ");
        gets(a);

  // scan the string to separate each word

        for(i  = 0; a[i] != '\0'; i++)
        {
                if(a[i] != ' ')
                        word[j++] = a[i];
                else
                {
                        b[k++] = word[0];
                        b[k++] = '.';
                        j = 0;
                }
        }

        word[j] = b[k] = '\0';

  // Concatenate last word in name as it is with string b

        strcat(b,word);

  // print the modified string b

        printf("\n%s",b);

        return(0);
}
```