Experiment-6
Stacks

(Duration: 120 mins)

Author: Anıl Karatay
anilkaratay@iyte.edu.tr

Purpose: Stacks are the data structures consisting of a pointer and a list of items. In this lab, a stack data structure will be designed and implemented.

Introduction

A stack is a linear data structure that serves as a collection of elements. There are three main operations that are push, pop and peek.

- Push: Add an element to stack
- Pop: Remove the top element from the stack
- Peek: Return the top item without changing the stack

The order in which the elements come off a stack gives rise to its alternative name, last in first out (LIFO). In Figure 1, you can see a simple representation of push and pop operations.

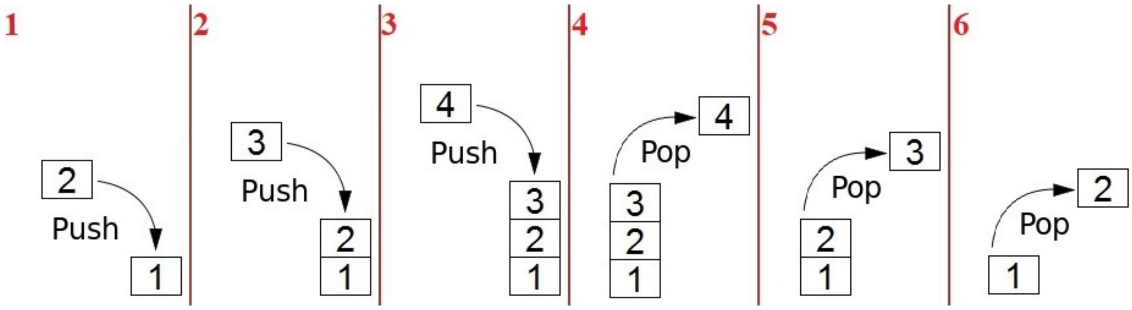


Figure 1: Push and pop operations

Problem Statement

A palindrome is a word, number or phrase which reads the same backward as forward. Some examples of palindromic words are redivider, deified, radar, level, reviver, racecar, madam, refer etc. You are asked to write a C program that counts the number of palindromic words in a given sentence. The program should also find the order of the palindromic words. As an example:

What did you do to my eye madam
Output:
Word 2 is a palindrome
Word 7 is a palindrome
Word 8 is a palindrome
Total number of palindromic words: 3

For the sake of simplicity, the user is allowed to write only alphanumeric characters (letters and numbers). Punctuation and other non-alphanumeric characters are not allowed.

Lab Procedure

- Your program should ask to the user a sentence without any punctuation. You can use the line below to include the white space. The maximum size of the array can be defined as a global variable, i.e 100.

```
scanf("%[^\n]s",&input);
```

- Declare the stack as a global structure as given below.

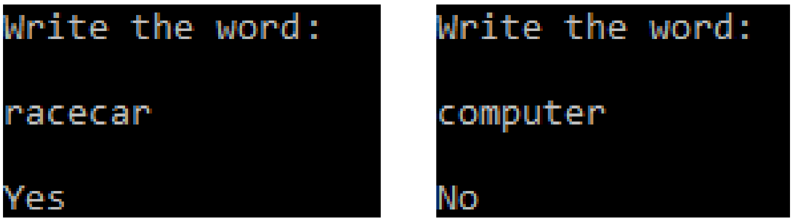
```
struct Stack{
char entry[MAX];
int count;
};
```

- Initialize the stack **(5 pts.)**. Write the stack functions that push() **(10 pts.)** and pop() **(10 pts.)** for this program. Be sure that the stack is not full during the push operation, and it is not empty during the pop operation **(5 pts.)**. If necessary, a stack pointer can also be written as an argument.

```
void initialize();//initialize the count as 0 or -1
void push(char c);//insert new character into the stack
char pop();//remove and return the top item
```

- Write a function that returns 1 if the input word is a palindrome. This function must be written using the push and pop operations. Otherwise it will not be evaluated **(20 pts.)**. Call this function at least once in the main, and show if it works for a single word. Printing commands must take place in main.

```
int checkPalindrome(char *inputWord);//Check whether the input word is a palindrome or not
```



```
Write the word:

racecar

Yes
```

```
Write the word:

computer

No
```

Figure 2: Check the checkPalindrome() function in main

- Write a function that counts the number of palindromic words in a given sentence **(20 pts.)**. The function should also indicate the order of the palindromic words **(20 pts.)**. For instance, assume that the 5th and 7th words of a 10-word sentence are palindrome. The output is like:

  Word 5 is a palindrome
  Word 7 is a palindrome
  Total number of palindromic words: 2

  The function's type casting and arguments are entirely up to you, but any print operation within the function is **prohibited**. All printing operations should be done in main. Note that, checkPalindrome() function must be called inside this function.

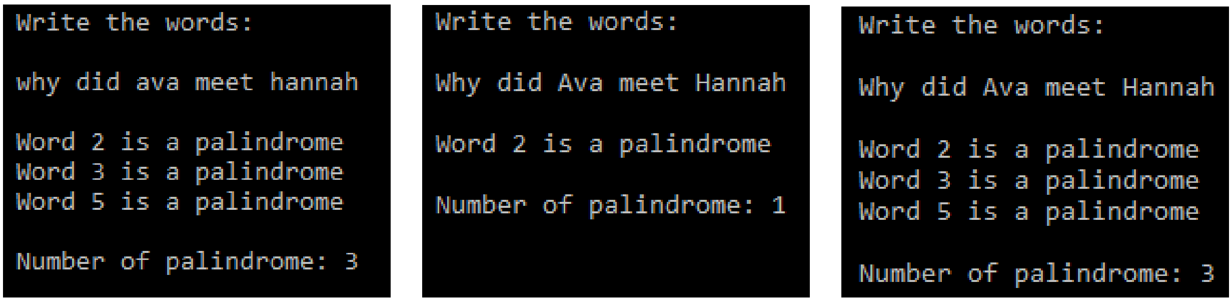
  Hint: You may use the white-space character to define the boundaries of the words. ASCII code of white-space is 32. You can also use ’ ’ to represent this. Also, end of the sentence is ’\0’.

- Print the results in main **(10 pts.)**.

- **Bonus:** Write a function that converts all uppercase letters to lowercase **(10 pts.)**. In this way, the user can type upper or lower case letters. You can specify the type of the function as you wish.

```
uppertoLower(char *input)//Convert uppercase to lowercase
```

Hint: Uppercase **A** has ASCII value 65 in decimal and for **Z**, the value is 90. Lowercase **a** has ASCII value 97 in decimal, and for **z**, the value is 122 in decimal. You are also free to use predefined functions in C libraries.



```
Write the words:

why did ava meet hannah

Word 2 is a palindrome
Word 3 is a palindrome
Word 5 is a palindrome

Number of palindrome: 3
```

(a)

```
Write the words:

Why did Ava meet Hannah

Word 2 is a palindrome

Number of palindrome: 1
```

(b)

```
Write the words:

Why did Ava meet Hannah

Word 2 is a palindrome
Word 3 is a palindrome
Word 5 is a palindrome

Number of palindrome: 3
```

(c)

Figure 3: Some sample outputs (a) Expected output without considering the bonus part (b) Expected erroneous output if the bonus part is not written (c) Correct output by including the bonus part