**Experiment No-09**

**Title: -**

stack. C++ program for a string of character that‘s the same forward and backward using

**Problem Statement: -**

A palindrome is a string of character that‘s the same forward and backward. Typically, punctuation, capitalization, and spaces are ignored. For example, “Poor Dan is in a droop” is a palindrome, as can be seen by examining the characters “poor danisina droop” and observing that they are the same forward and backward. One way to check for a palindrome is to reverse the characters in the string and then compare with them the original- in a palindrome, the sequence will be identical. Write C++ program with functions-

a) To print original string followed by reversed string using stack

b) To check whether given string is palindrome or not

**Theory-**

*Basic Concept of Stack:-*

A stack is an ordered list in which all the insertion and deletion are made at one end called as top. If we have to make stack elements of 10, 20, 30, 40, 50, 60 then 10 will be the bottommost element and 60 will be the topmost element in the stack. As stack is shown in figure

Representation of stack in C++:-

struct stack

{ int data[10]; int top;

}s;

***Stack Operations:-***

Basically there are two important stack operation 1) push and 2) pop

Performing Push Operation means we are inserting elements onto the stack . and Pop Operation means we are removing the element from the stack.

Before pushing we need to check stack full condition and before performing pop operation we need to check stack empty condition.

**Stack Empty Operation:-**

Initially stack is empty. At that time the top should be initialized to -1 or 0. If we set top to -1 initially then the stack will contains the elements from 0th position and if we set top to 0 initially, the element will be stored from 1st position, in the stack. Elements may be pushed onto the stack and there may be a case that all the elements are removed from the stack. Then the stack becomes empty. Thus whenever top reaches to -1 we can say that the stack is empty.

int stempty()

{

if(s.top==-1)

return 1;

else

return 0;

}

**Stack Full Operation:-**

In the representation of stack using array, size of the array means size of the stack. As we go on inserting the elements the stack gets filled with the elements. So it is necessary before inserting the elements to check whether the stack is full or not. Stack full condition is achieved when stack reaches to the maximum size of array – 1

int stfull()

{

if(s.top>=size -1)

return 1;

else

}

return 0;

Thus stfull is Boolean function if stack is full it return 1 otherwise it return 0.

**The push and Pop Operation:-**

1) push Operation

void push(int Item)

{ st.top++; / \* top pointer is set to next location\*/

st.s[st.top] =item;/\* placing the element at that location \*/

}

2) pop operation

int pop( )

{

int item;

item = st.s[st.topl; st.top--;

return(item);

}

In the choice of pop- it invokes the function isempty to determine whether the stack is

empty or not. If it is empty, then the function generates an error as stack underflow. If not, then pop function returns the element which is at the top of the stack.

Algorithm:-

1) Start

2) Declare the variables such as a, top, i, max

3) Take a string as an input from user store it in array a

4) Remove quotations, spaces etc. from a string **a** by giving call to convert function

4) Call function push ( ) to push all characters of string to stack.

5) Call function of palindrome ( ) to check string is palindrome or not

6) Call function reverse ( ) to print reverse of a given string.

7) Stop

Algorithm of push()

1. Increment top
2. put given character x on the top of a stack i.e. at a[top]
3. Put ‘\0’ at a[top+1] to indicate end of a string.

Algorithm of pop()

1. Take out a character which is present on the top of a stack a and store it in x

i.e. x= a[top]

1. Decrement top by 1
2. Return x as poped character.

Algorithm of reverse()

1. Initialize j=0
2. i=top
3. Take out a character which is present on the top of a stack and store it array str

i.e. str[j]= a[i]

1. Decrement top by 1 increment j by 1
2. Repeat step2 to 4 till top!=-1
3. Print str as a reversed string.

Algorithm of palindrom()

1. Initialize j=0.
2. i=top
3. Copy a[i] into str[j]
4. Decrement top by 1 increment j by 1
5. Repeat step 2 to 4 till top!=-1
6. Compare strings **str** and **a** if they match print that string is a palindrom

Algorithm of convert()

1. For all characters of string **a** do following steps
2. If character from string is uppercase, convert it into lowercase and store in str

Else store it in str as it is

1. If character is any other character than alphabet then ignore it

Conclusion:

By this way, we can learn how to implement stack, push and pop operations in deep.