**VARUN KUMAR**

**2K19-IT-140**

**DS LAB – 6**

11. Write a program to check weather given . string is Palindrome or not using DEQUEUE.

# include<stdio.h>

#include <string.h>

# define MAX 30

int deque\_arr[MAX];

int left = -1;

int right = -1;

/\*Begin of insert\_right\*/

void insert\_right(char added\_item)

{

if((left == 0 && right == MAX-1) || (left == right+1))

return;

if (left == -1) /\* if queue is initially empty \*/

{ left = 0;

right = 0;}

else

if(right == MAX-1) /\*right is at last position of queue \*/

right = 0;

else

right = right+1;

deque\_arr[right] = added\_item ;

}

/\*End of insert\_right\*/

/\*Begin of insert\_left\*/

void insert\_left(char added\_item){

if((left == 0 && right == MAX-1) || (left == right+1))

return;

if (left == -1)/\*If queue is initially empty\*/

{ left = 0;

right = 0; }

else

if(left== 0)

left=MAX-1;

else

left=left-1;

deque\_arr[left] = added\_item ; }

/\*End of insert\_left\*/

/\*Begin of delete\_left\*/

void delete\_left()

{ if (left == -1)

return ;

if(left == right) /\*Queue has only one element \*/

{ left = -1;

right=-1; }

else

if(left == MAX-1)

left = 0;

else

left = left+1;

}

/\*End of delete\_left\*/

/\*Begin of delete\_right\*/

void delete\_right()

{if (left == -1)

return ;

if(left == right) /\*queue has only one element\*/

{ left = -1;

right=-1; }

else

if(right == 0)

right=MAX-1;

else

right=right-1; }

/\*End of delete\_right\*/

int front(){

return deque\_arr[left];

}

int back(){

return deque\_arr[right];

}

bool empty(){

if(left==-1 and right==-1) return 1;

else return 0;

}

/\*Begin of main\*/

int main()

{

char s[20];

printf("Enter the string : ");

gets(s);

for(int i=0 ;i<strlen(s);i++){

insert\_right(s[i]);

}

bool palindrome=1;

while(!empty()){

char f=front();

char b=back();

if(f!=b) palindrome=0;

delete\_left();

delete\_right();

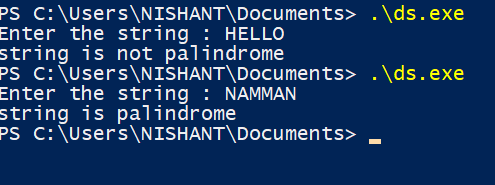
}

if(palindrome) printf("string is palindrome\n");

else printf("string is not palindrome\n");

return 0;

}



# 12: Implement Tower of Hanoi Problem using Stack

**// C Program for Tower of Hanoi**

**#include <stdio.h>**

**#include <math.h>**

**#include <stdlib.h>**

**#include <limits.h>**

**// A structure to represent a stack**

**struct Stack**

**{**

**unsigned capacity;**

**int top;**

**int \*array;**

**};**

**// function to create a stack of given capacity.**

**struct Stack\* createStack(unsigned capacity)**

**{**

**struct Stack\* stack = (struct Stack\*) malloc(sizeof(struct Stack));**

**stack -> capacity = capacity;**

**stack -> top = -1;**

**stack -> array =**

**(int\*) malloc(stack -> capacity \* sizeof(int));**

**return stack;**

**}**

**// Stack is full when top is equal to the last index**

**int isFull(struct Stack\* stack)**

**{**

**return (stack->top == stack->capacity - 1);**

**}**

**// Stack is empty when top is equal to -1**

**int isEmpty(struct Stack\* stack)**

**{**

**return (stack->top == -1);**

**}**

**// Function to add an item to stack. It increases**

**// top by 1**

**void push(struct Stack \*stack, int item)**

**{**

**if (isFull(stack))**

**return;**

**stack -> array[++stack -> top] = item;**

**}**

**// Function to remove an item from stack. It**

**// decreases top by 1**

**int pop(struct Stack\* stack)**

**{**

**if (isEmpty(stack))**

**return INT\_MIN;**

**return stack -> array[stack -> top--];**

**}**

**//Function to show the movement of disks**

**void moveDisk(char fromPeg, char toPeg, int disk)**

**{**

**printf("Move the disk %d from \'%c\' to \'%c\'\n",**

**disk, fromPeg, toPeg);**

**}**

**// Function to implement legal movement between**

**// two poles**

**void moveDisksBetweenTwoPoles(struct Stack \*src,**

**struct Stack \*dest, char s, char d)**

**{**

**int pole1TopDisk = pop(src);**

**int pole2TopDisk = pop(dest);**

**// When pole 1 is empty**

**if (pole1TopDisk == INT\_MIN)**

**{**

**push(src, pole2TopDisk);**

**moveDisk(d, s, pole2TopDisk);**

**}**

**// When pole2 pole is empty**

**else if (pole2TopDisk == INT\_MIN)**

**{**

**push(dest, pole1TopDisk);**

**moveDisk(s, d, pole1TopDisk);**

**}**

**// When top disk of pole1 > top disk of pole2**

**else if (pole1TopDisk > pole2TopDisk)**

**{**

**push(src, pole1TopDisk);**

**push(src, pole2TopDisk);**

**moveDisk(d, s, pole2TopDisk);**

**}**

**// When top disk of pole1 < top disk of pole2**

**else**

**{**

**push(dest, pole2TopDisk);**

**push(dest, pole1TopDisk);**

**moveDisk(s, d, pole1TopDisk);**

**}**

**}**

**//Function to implement TOH puzzle**

**void tohIterative(int num\_of\_disks, struct Stack**

**\*src, struct Stack \*aux,**

**struct Stack \*dest)**

**{**

**int i, total\_num\_of\_moves;**

**char s = 'S', d = 'D', a = 'A';**

**//If number of disks is even, then interchange**

**//destination pole and auxiliary pole**

**if (num\_of\_disks % 2 == 0)**

**{**

**char temp = d;**

**d = a;**

**a = temp;**

**}**

**total\_num\_of\_moves = pow(2, num\_of\_disks) - 1;**

**//Larger disks will be pushed first**

**for (i = num\_of\_disks; i >= 1; i--)**

**push(src, i);**

**for (i = 1; i <= total\_num\_of\_moves; i++)**

**{**

**if (i % 3 == 1)**

**moveDisksBetweenTwoPoles(src, dest, s, d);**

**else if (i % 3 == 2)**

**moveDisksBetweenTwoPoles(src, aux, s, a);**

**else if (i % 3 == 0)**

**moveDisksBetweenTwoPoles(aux, dest, a, d);**

**}**

**}**

**// Driver Program**

**int main()**

**{**

**// Input: number of disks**

**unsigned num\_of\_disks = 4;**

**struct Stack \*src, \*dest, \*aux;**

**// Create three stacks of size 'num\_of\_disks'**

**// to hold the disks**

**src = createStack(num\_of\_disks);**

**aux = createStack(num\_of\_disks);**

**dest = createStack(num\_of\_disks);**

**tohIterative(num\_of\_disks, src, aux, dest);**

**return 0;**

**}**

