DATA - STRUCTURES

HOME - ASSIGNMENT - 2

CODE : 20IT3303

Submitted By:

M.J.N.V. Sai

208W1A12A0

IT-B

Submitted To:

V. Radhesyam

Assistant - Professor

IT DEPARTMENT

"DATA-STRUCTURES"

M.J.N.V.Sai
208W1A1240
IT-B

HOME-ASSIGNMENT-2

Implement the following operations of a Queue using Stacks.

1) EnQueue(x) : Insert an item x to the Rear of Queue.

2) DeQueue (x) : Remove an item from front of Queue.

3) Peek () : Get the front item.

4) Empty() : Return whether the Queue is empty.

problem note:

1) You must use only standard operations of stack : which means only push, POP, Peek, from toP, size, isempty operations are valid.

2) Depending on your language, stack may not be supported natively. You may simulate a stack by using a list or Dequeue (Double-Ended-Queue), as long as yu use only standard operations of a stack.

3) You may assume that all operations are valid (For ex, no POP or peek operations will be called on an empty Queue).

→ Queue can be Implemented by Using Stacks
Required : 2 stacks.

→ It can be Implemented by EnQueue & DeQueue operations.

Algorithm : Queue Using stacks

step 1 : Initilize stack1 and stack2 by maki

Their Top of stack To -1 { TOP1 = TOP2 = -

step 2 : To perform an EnQueue Operation.

step 3 : push All elements To stack 1 from To stack 2

step 4 : Push the New element into the stack 2

step 5 : pop All the elements from stack2 To stac

step 6 : Now, POP and Return the element fro stack1.

program :

```
# include <stdio.h>
# include <stdlib.h>
// Declarations
void push1(int);
void push2 (int);
int pop1();
int pop2();
void enqueue();
void dequeue();
void display();
 void create();
int stack1[100], stack2[100], top1 = -1, top2 = -1, count = 0;
void create() { top1 = top2 = -1; }
// To push an Element To a stack;
void push1 (int element) { stack1[++top1] = element; }
// To pop an element from a stack;
int pop1 () {    return(stack1[top1--]); }
// To push an element To a stack;
void push2 (int element) { stack2 [++top2] = element; }
// To pop an Element from a stack;
void pop2 () {    return (stack2 [top2--]); }
// To Enqueue an value into Queue using stack;
void Enqueue ()
{
    int data, i;
    printf(" enter the data:   ");
    scanf("%d", &data);
    push1(data);
    count++;
 }
```

```
// To DeQueue an Value from Queue using stack.

void dequeue()
{
    int i;
    For (i=0; i<= count ; i++)
    {
        Push2 (Pop1());
    }
    Pop2();
    Count --;
    For (i=0; i<= count; i++) { Push1(Pop2()); }
}
void display()
{
    int i;
    If (top1 = = -1) { printf(" \n Empty queue \n"); }
    else
    {
        printf (" \n Queue elements :    ");
        For (i=0; i<=top1; i++)
        {
            printf (" %d ", stack1[i]);
        }
        printf("\n");
    }
}
```

```
int main()
{
    int choice;
    printf ("\n 1. Enqueue ");
    printf ("\n 2. DeQueue ");
    printf ("\n 3. Display ");
    printf ("\n 4. Exit \n");
    create ();
    while (1)
    {
        printf ("\n Enter your choice :    ");
        scanf ("%d", &choice);
        switch (choice)
        {
            case 1 :
                    Enqueue ();
                    Break;
            case 2 :
                    DeQueue ();
                    Break;
            case 3 :
                    Display ();
                    Break;
            case 4 :
                    Exit (0);
                    Break;
        Default :
                printf ("\n invalid choice \n");
        }
    }
    return 0;
}
```

OUTPUT :

```
1. EnQueue
2. DeQueue
3. Display
4. Exit

Enter your choice : 1
Enter The data : 15

Enter your choice : 1
Enter The data : 30

Enter your choice : 1
Enter The data : 45

Enter your choice : 3

Queue Elements : 15  30  45
```

```
Enter your choice : 2

Enter your choice : 3
Queue Elements :   30   45

Enter your choice : 1
Enter the data : 60

Enter your choice : 3
Queue Elements  :  30   45   60

Enter your choice : 4
```

RESULT: Successfully Executed the program