Yash Mahajan SE IT B 04 19106 Page: Date: / /

Experiment 2

1. Aim :- Write a program to create linked list implementation of stack and queue.

2. Objectives :- After study of this experiment, the student will be able to
   - Understand how to create a stack and queue
   - Implement an algorithm using computer to solve the given problem

3. Outcomes :-
   - Developing algorithms for various problems on basic concepts and principles of various stacks, linked lists and queues.

4. Prerequisite :- stack, queue and its operations.

5. Requirements :- PC and Turbo C compiler version 3.0.

Yash Mahajon 04 SE IT-B 191061

Page :
Date : / /

What is a stack ?

A stack is a linear data structure which uses a particular order in which operations are carried out. Elements of a stack are added or removed only from one end, which is called the TOP. Hence a stack is called a LIFO (Last in first out : data structure as the element that was inserted last is the first one to be taken out.

Operation on Stack :-

Push operation :-

The process of putting an element in a stack is called as push. The new element is added at the topmost position of the stack. The steps are :-

```
Step 1 : IF TOP = MAX-1
             Print "Overflow"
Step 2 :-    Goto Step 4
         [END IF]
Step 2 : SET TOP = TOP+1
Step 3 :- SET STACK[TOP] = VALUE
Step 4 :- END.
```

POP operation :-

The pop operation is used to delete the topmost element in a stack. The steps involved are :-

```
Step 1 : IF TOP = = NULL
             PRINT "UNDERFLOW"
             Goto Step 4
         [END IF]
Step 2 :- TOP = TOP-1 SET VAL = STACK[TOP]
Step 3 :- SET TOP = TOP-1.   Step 4 : End.
```

Yash Mahajan 04 SE IT-B 191061

Page:
Date: / /

What is a queue :-

A queue is a linear data structure. It follows FIFO principle (First in First out) data structure in which the element that is inserted first is the first one to be taken out. The elements in a queue are added at one end called REAR and removed from the other end called FRONT.

Operations on a queue :-

- enque () :- add (store) an item to the queue
- dequeue () :- remove (access) an item from the queue.

peek () - gets the element at the front of the queue without deleting.

isfull() :- checks if the queue is full.

is empty() :- checks if the queue is full.

7. Laboratory Exercise

A. Procedure: Stack

Program:- Stack Implementation of Linked List

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<malloc.h>

struct stack {
   int data;
   struct stack *next;
};

struct stack * top = NULL;

struct stack *push(struct stack *top, int val);
struct stack *pop(struct stack *top);
void peak(struct stack *top);
void display(struct stack *top);


int main(){

   int ch, val;

   do{   printf("\nEnter\n1 to Push \n2 to Pop \n3 to Peek \n4 to display
\n5 to exit");
       printf("\nEnter your choice:");
       scanf("%d",&ch);
       switch (ch)
       {
       case 1:printf("\nEnter the element:");
           scanf("%d",&val);
           top = push(top,val);
           display(top);
```

```
            break;
        case 2:top = pop(top);
            if (top != NULL){
               display(top);
            }
            break;
        case 3:
            peak(top);
            break;
        case 4:
            display(top);
            break;

        default:
            break;
        }
    } while (ch != 5);

    return 0;
}

struct stack *push(struct stack *top, int val){
    struct stack *new_node;
    new_node = (struct stack *)malloc(sizeof(struct stack));
    new_node -> data = val;

    if (top != NULL){
        new_node -> next = top;
        top = new_node;
    }
    else{
        new_node -> next = NULL;
        top = new_node;
    }
    return top;
}
```

```
struct stack *pop(struct stack *top){
   if (top !=NULL){
       struct stack *ptr;
       ptr = top;
       printf("\n The value being deleted is : %d",ptr->data);
       top = top -> next;
       free(ptr);
   }
   else{
       printf("\n UNDERFLOW Stack is empty");
   }
   return top;
}

void peak(struct stack *top){

   if (top !=NULL){
       struct stack *ptr;
       ptr = top;
       printf("\n Top: %d",ptr->data);
   }
   else{
       printf("\n UNDERFLOW Stack is empty");
   }
}

void display(struct stack *top){
   if (top !=NULL){
       struct stack *ptr;
       ptr = top;
       printf("\nStack: ");
        while(ptr != NULL){
           printf("%d ", ptr -> data);
           ptr = ptr -> next;
       }
   }
   else{
```

```
        printf("\n UNDERFLOW Stack is empty");
    }
}
```

B. Result/Observation/Program code:

```
yashmmahajan19@penguin:~/Work/DSA/exp 2$ ./prog1

Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:1

Enter the element:10

Stack: 10
Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:1

Enter the element:20

Stack: 20 10
Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:4

Stack: 20 10
Enter
1 to Push
2 to Pop
```

```
3 to Peek
4 to display
5 to exit
Enter your choice:4

Stack: 20 10
Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:3

 Top: 20
Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:2

 The value being deleted is : 20
Stack: 10
Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:3

 Top: 10
Enter
1 to Push
```

```
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:2

 The value being deleted is : 10
Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:3

 UNDERFLOW Stack is empty
Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:2

 UNDERFLOW Stack is empty
Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:4

 UNDERFLOW Stack is empty
```

```
Enter
1 to Push
2 to Pop
3 to Peek
4 to display
5 to exit
Enter your choice:5
yashmmahajan19@penguin:~/Work/DSA/exp 2$
```

C. Procedure: Queue

Program:- Queue implementation of Linked List

```c
#include <stdio.h>
#include<conio.h>
#include<stdlib.h>
#include <malloc.h>

struct node{
   int data;
   struct node *next;
};
struct queue{
   struct node *front;
   struct node *rear;
};

struct queue *q = (struct queue*)malloc(sizeof(struct queue));

void create_queue(struct queue *);
struct queue *enqueue(struct queue *,int);
struct queue *dequeue(struct queue *);
void display(struct queue *);

int main(){

   int val, ch;
```

```
   create_queue(q);
   do{
       printf("\nEnter\n1 to Insert \n2 to Delete \n3 to Display \n4 to
Exit");
       printf("\nEnter your choice:");
       scanf("%d",&ch);
       switch (ch)
       {
       case 1:printf("\nEnter the element:");
           scanf("%d",&val);
           q = enqueue(q,val);
           display(q);
           break;
       case 2:
           q = dequeue(q);
           if (q -> front != NULL)
           {
              display(q);
           }
           break;
       case 3:
           display(q);
           break;
       default:
           break;
       }
   } while (ch != 4);
   return 0;
}

void create_queue(struct queue *q){
   q -> rear = NULL;
   q -> front = NULL;
}

struct queue *enqueue(struct queue *q,int val){
   struct node *new_node;
```

```
    new_node = (struct node*)malloc(sizeof(struct node));
    new_node -> data = val;
    if(q -> front != NULL){
        q -> rear -> next = new_node;
        q -> rear = new_node;
        q -> rear -> next = NULL;
    }
    else{
        q -> front = new_node;
        q -> rear = new_node;
        q -> front -> next = q -> rear -> next = NULL;
    }
    return q;
}

struct queue *dequeue(struct queue *q)
{
    struct node *new_node;
    new_node = q -> front;
    if(q -> front != NULL){
        q -> front = q -> front -> next;
        printf("\n The value being deleted is : %d", new_node -> data);
        free(new_node);
    }
    else
         printf("\n UNDERFLOW Queue is empty");
    return q;
}

void display(struct queue *q){
    struct node *new_node;
    new_node = q -> front;
    if(new_node != NULL){
        printf("\nQueue: ");
        while(new_node!=q -> rear){
            printf("%d ", new_node -> data);
            new_node = new_node -> next;
```

```
        }
        printf("%d ", new_node -> data);
    }
    else
        printf("\n UNDERFLOW QUEUE IS EMPTY");
}
```

D. Result/Observation/Program code:

```
yashmmahajan19@penguin:~/Work/DSA/exp 2$ ./prog2

Enter
1 to Insert
2 to Delete
3 to Display
4 to Exit
Enter your choice:1

Enter the element:10

Queue: 10
Enter
1 to Insert
2 to Delete
3 to Display
4 to Exit
Enter your choice:1

Enter the element:20

Queue: 10 20
Enter
1 to Insert
2 to Delete
3 to Display
4 to Exit
Enter your choice:3

Queue: 10 20
Enter
1 to Insert
2 to Delete
3 to Display
4 to Exit
Enter your choice:2
```

```
 The value being deleted is : 10
Queue: 20
Enter
1 to Insert
2 to Delete
3 to Display
4 to Exit
Enter your choice:2

 The value being deleted is : 20
Enter
1 to Insert
2 to Delete
3 to Display
4 to Exit
Enter your choice:2

 UNDERFLOW Queue is empty
Enter
1 to Insert
2 to Delete
3 to Display
4 to Exit
Enter your choice:3

 UNDERFLOW QUEUE IS EMPTY
Enter
1 to Insert
2 to Delete
3 to Display
4 to Exit
Enter your choice:4
yashmmahajan19@penguin:~/Work/DSA/exp 2$
```

Yash Mahajan 04 SEIT-B 191061

Page: Date: / /

Questions

1. Applications of Stack ~~are~~ :-

Applications of stak are :-

- Reversing a list
- Parentheses checker
- Conversion of an infinix expression into a postfix expression.
- Evaluation of a postfix expression.
- Conversion of an infinix expression into a prefix expression
- Evaluation of a prefix expression.
- Recursion.
- Tower of Hanoi

(ii). Application of ~~Queue~~ Queue :-

Applications of Queue are :-

- Queues are widely used as waiting lists for a single shared resource like printer, disk, CPU.
- Queues are used to transfer data asynchronously between two processes eg ~~pot~~ pipes, file IO, sockets.
- Queues are used as buffers on MP3 players and portable CD players, iPod playlist.
- Queues are used in Playlist of jukebox to add songs to the end, play from the front of the list.
- Queues are used in operating system for handeling interrupts.

9. Conclusion:

Yash Mahajan 04 SEIT-B 191061

Page: 6
Date: 1 1

Conclusion:-

The aim of the experiment was achieved by first understanding how operations like push, pop for stack and enque, dequeue for queue are executed. Using these principles implementation of linked list as stack and que was carried out.

Stacks and queues are both commonly used data structures to store and retrive data dynamically. By studying this user can understand about the various operations and applications of stacks and queues.

Stacks and queus are linear data structures. Their various applications are reversing of a string, recurssive descent parsing, breadth-first search, asynchronous data transfer etc.