Assignment No. 2.

Page No.
Date

Q.1) Explain representation and implementation of queue and stack using sequential and linked allocation.

→

① Stack :

- Representation of Stack in Memory

The stack can be implemented into two ways :

Using Arrays (static implementation)
Using pointer (Dynamic implementation)

i) Using Arrays :

(i) we define a one dimensional array of specific size and insert or delete the values into that array by using LIFO principle with the help of a variable called "top".

(ii) Initially, the top is set to -1.

(iii) Whenever, we want to insert a value into the stack, increment the top value by one and then insert. Whenever we want to delete a value from the stack, then delete the top value and decrement the top value by one.

Advantages of Array implementation :

① Easy to implement
② Memory is saved as pointers are not involved.

Disadvantage of array implementation:

i) It is not dynamic.

ii) It doesn't grow and shrink depending on needs at runtime.

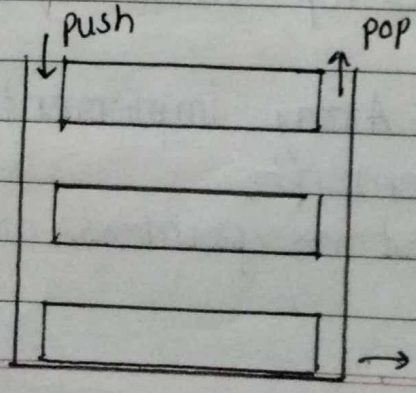
ii) Using Linked List

(i) The major problem with the stack implemented using an array is, it works only for a fixed number of data values.

(ii) The stack implemented using linked list can work for an unlimited number of values. That means, stack implemented using linked list works for the variable size of data.

(iii) In linked list implementation of a stack, every new element is inserted as 'top' element. Whenever we want to remove an element from the stack, simply remove the node which is pointed by 'top' by moving 'top' to its previous node in the list.

(iv) The next field of the first element must be always NULL.



Page No.
Date

(2) Queue :

The queue can be implemented into two ways
i) Using arrays (Static implementation)
ii) Using pointer (Dynamic implementation).

i) Using arrays.

(i) Queue is represented in memory using linear array.
(ii) Let QUEUE is a array, two pointer variables called FRONT and REAR.
(iii) The pointer variable FRONT contains the location of the element to be removed or deleted.
(iv) The pointer var REAR contains the location of last element to be inserted.
(v) The condition FRONT = NULL indicates that queue is empty.
(vi) REAR = N-1 indicates that queue is full.

Program :

```
void insert (int queue [], int max, int front, int rear, int item){
    if (rear +1 == max) {
        printf ("overflow");
    } else {
        if (front == 1 && rear == -1){
            front = 0, rear = 0;
        } else {
            rear = rear + 1;
        }
    } queue [rear] = item; }}
```

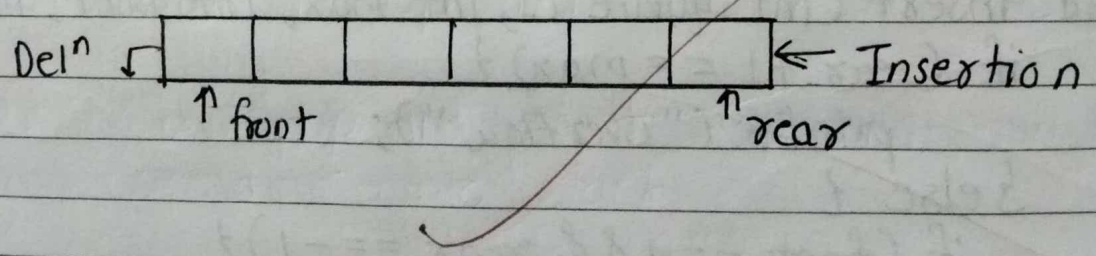
Page No.
Date

Queue using pointers.

```
void insert (int queue [], int * max, int * front,
             int * rear, int * item){

    if (*rear +1 = *max){
        printf ("overflow");
    }
    else {
        if (*front == 1 && *rear == -1){
            *front =0, *rear =0;
        }
        else {
            * rear = * rear +1;
        }
    queue [*rear] = * item;
}
}
```

Queue Representation



Page No.
Date

Q.2 Explain Queue and its operations.

→

A queue is an ordered collection of items where an item is inserted at one end called "rear" and an existing item (1st) is removed at the other end called the "front".

i) Queue follows the First in First out (FIFO) principal.
ii) For example, a queue of people waiting at a bus stop where each new person who comes takes his or her place at the end of line, and when the bus comes, the people at the front of line board first.

Operations of Queue

i) Queue() :
ii) Enqueue() : adds or store an element at the end of the queue.
iii) Dequeue() : Removal of elements from the queue.
iv) front() : Returns the value present at front node without deleting it.
v) rear() : This operation returns the element at the rear end without deleting it.

vi) isFull(): checks whether queue is full or not.
vii) size() : returns the size of queue.

Page No.
Date

Q.3 Explain Step by Step procedure of infix to postfix conversion of stack. Give an example.

→

Step 1: Scan the expression from left to right
Step 2: Print the operands as they arrive.
Step 3: If the stack is empty or contains '(' parenthesis on the top, push the incoming operator on to the stack.
Step 4: If the incoming symbol is '(' then push into the stack.
Step 5: If the incoming symbol is ')' then pop the stack and print the operator until left parenthesis is found.
Step 6: If the incoming symbol has higher precedance than the top then push it into the stack.
Step 7: If the incoming symbol has lower precedance than the top then pop & print top then test incoming operator against new top of the stack.
Step 8: If the incoming symbol has equal precedance with the top then use Associative rule.

(a) L→R: Then pop and print top and push incoming operator into stack.

(b) R→L: Push into the stack.

Page No.
Date

Example. A - B / C * D + E

Symbol	postfix	Stack
A	A	.
-	A	-
B	AB	-
/	AB	- /
C	ABC	- /
*	ABC/	- *
D	ABC/D	- *
+	ABC/D* -	+
E	ABC/D* - E	+
	ABC/D* - E +	.

ABC/D*-E+

Q.4 List and explain the application of Stack.

→

i) It is used to reverse a word. You push a given word to stack - letter by letter and then pop letter from the stack.

ii) Undo mechanism in text editor.

iii) Backtracking : This is a process when you need to access the most recent data element in a series of elements. Once you reach a dead end, you must backtrack

iv) Conversion of decimal number to binary.

v) To solve tower of Hanoi

vi) Runtime memory management.

vii) Quick sort

Q.5) List and explain applications of Queue.

→

i) Web servers : Web servers use queues to manage incoming requests from clients. Requests are added to queue as they received and processed as they were received in order.

ii) Printer Queue: In printing systems, queues are used to manage the order in which print jobs are processed. Jobs are added to queue as they are submitted and printer processes them in the order they were received.

iii) Operating Systems : Operating systems often use queues to manage processes and resources.

iv) Task Scheduling

v) Event Handling

vi) Traffic Management

vii) Applied as buffers on MP3 players, etc.

Page No.
Date

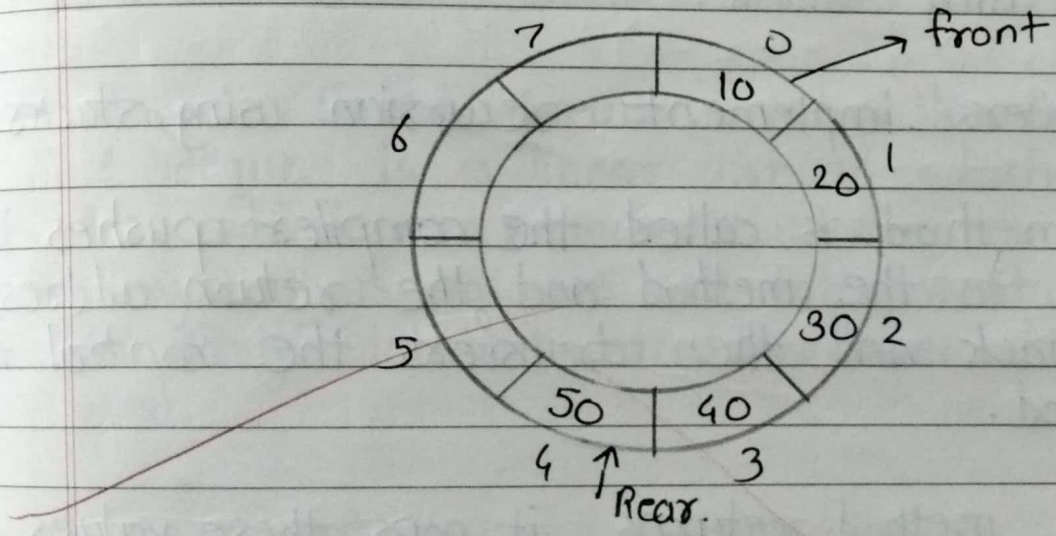
Q.6) List the types of queue & explain circular queue with example.

→

Types of queue :
1) Circular Queue
2) Priority Queue
3) Simple Queue.
4) De-queue (Double Ended Queue).

Circular Queue.

A circular queue is a queue in which all nodes are treated as circular such that the last node follows the first node.



Algo ?

Suppose Q is a queue array of 8 elements. Insertion and deletion operation can be performed on circular. basis.

Q.7) Explain Recursion in Stack.

→

Recursion : A programming technique in which a function calls itself.

i) Recursion can be used to find the n^{th} term.

ii) If a loop is used, the method cycles around the loop n times, adding n to the total the first time, n-1 the second time and so on, down to 1, quiting loop when n becomes 0.

iii) Memory is used to store all the intermediate arguments and return values ~~of the return point.~~ on the internal stack.

Recursion using Stack.

i) Most compilers implement recursion using stacks.

ii) When a method is called the compiler pushes the arguments to the method and the return address on the stack and then transfers the control to the method.

iii) When the method returns, it pops these values off the stack.

iv) The general idea behind recursion is.

① To divide a problem into smaller pieces until reaching to solvable pieces.

② Then, to solve the problem by solving these small pieces of problem and merging the solutions to

each other.

v) Generally a recursive algorithm has two parts. The 1st part checks if the problem is solvable and solves it.

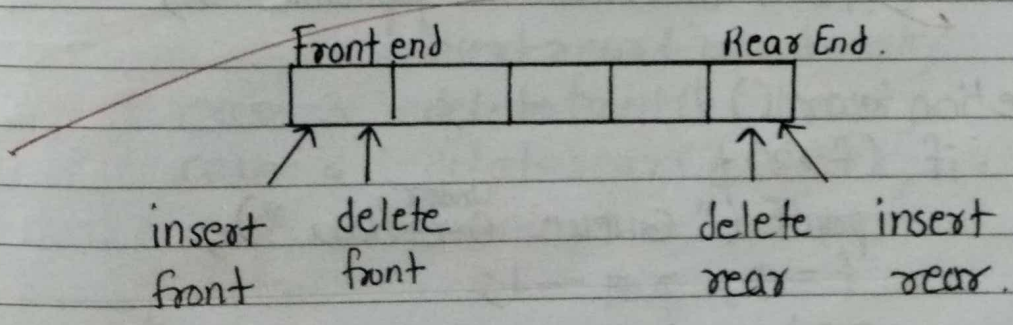
vi) The 2nd part that divides the problem into smaller pieces in each round.

vii) Here in the 2nd part we see that the same function calls itself many times to break the problem into smaller pieces.

Q.8) Explain Double Ended Queue.

→

i) Double Ended Queue is also known as Dequeue.
ii) A deque is a linear list in which elements can be added or removed at either end.
iii) Insertion is possible at both front and rear end. Similarly, deletion can be done at front and rear end.



Date

* The operations performed are:
a) insertFront()
b) insertRear()
c) DeleteFront()
d) deleteRear()
e) display().

Implementation of Double-ended queue.

```
# define size 5
int q[size];
int f=0, r=-1;

void insertfront(){
    if (f==0 && r==-1){
        q[++r] = item;
        return;
    }
    if (f!=0){
        q[--F] = item;
        return;
    }
    print("Front insertion not possible");
}
void deletionrear(){
    if (f>r){
        printf("Queue Underflow");
        f=0, r=-1;
        return;
    }
```

```
    printf ("Element deleted = %d \n", q[r--]);
}
void display (){
    int i;
    if (f>r){
        printf (" Q is empty - underflow");
        f = 0, r = -1;
        return;
    }
    if ("Contents of Queue are:");
    for (i=f; i<=r; i++)
        printf ("%d \n", q[i]);
}
void main (){
    int ch;
    {
    pf (" 1. Insert F  2. Insert R  3. Delete F  4. Delete R
        5. Display  6. Exit \n");
    pf ("Enter your choice");
    sf ("%d", &ch);
    switch (ch){
      case 1 : pf("Enter item\n"); sf ("%d", &item);
               insertfront (); break;
      case 2 : pf ("Enter item \n"); sf ("%d", &item);
               insertrear (); break;
      case 3 : deletefront(); break;
      case 4 : deleterear(); break;
      case 5 : display (); break;
      default : exit (0);
    }
  }
```

Q.9 Explain Priority Queue.

→

i) The priority queue is a special type of data structure in which items can be deleted or inserted based on priority.

ii) Always an element having highest priority is processed first

iii) If the elements in the queue are of the same priority, then the element which is inserted first into the queue is processed.

Applications :

i) Used in operating system for the design of Job scheduling Algorithms.

ii) Ascending Priority Queue : Here elements are inserted in any order. But while deleting an item from the queue, only the smallest element is removed first.

iii) Descending priority queue : Here also elements are inserted in any order. But while deleting an element the largest element deleted first.

Basic Operations :

i) insert or enqueue - add an item to the rear of the queue.

ii) remove or dequeue - remove an item from the front of the queue.

Priority Queue as ADT

i) size() - return the number of entries in PQ.
ii) isEmpty() - test whether PQ is empty
iii) min() - return (but not remove) the entry with the smaller key.
iv) insert(k, x) - insert value x with key k.
v) removeMin() - remove from PQ & return the entry with the smallest key.

Q.10) Write an algorithm to evaluate postfix using stack and execute your algorithm with postfix expression 10, 5, 6, *, +, 8 /. Show the intermediate stack content after each operation.

→

Step 1 : Create an empty stack.

Step 2 : Iterate through the postfix expression from left to right.

Step 3 : For each element in the postfix expression, ~~from~~ do the following.

1. If the element is an operand (a number), push it onto the stack.
2. If the element is an operator (+, /, -, etc) pop the top two elements from stack, apply the operator to these elements and push the result back onto the stack.

Step 4: After all elements have processed, the result of the expression will be single element left on the stack.

Symbol	Postfix	Stack
10		10
5		10 5
6		10 5 6
*	5*6	10 30
+	10 +30	40
8		40 8
/	40/8	5

Result = 5.

20/20

Aparant
14/09/2023