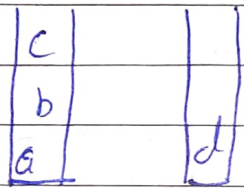
Name:- BREMDAM LUCAS ROLLNO:- 8953

Page No. Date

D.S - U.T - 1

Q1

(a) Ans :- (D) :- c d a b



s2.pop = c
s1.pop = d
s2.pop = b

stack 2 stack 1

~~In~~

In no way it is possible for a to come before b

(b) (A) List implementation with an array.

~~B~~

because with a position n it can be easily called as a[n-1]. as index is pos = 1;

Q2

State of stack and queue after end of first while loop

stack :- | 50 | 40 | 30 | 20 | 10 |

queue :- [ ] ↑ empty

State of stack and queue after end of second loop

stack s :- [ ] ↑ empty

queue :- | 50 | 40 | 30 | 20 | 10 |

NAME:- BRENDAN LUCAS ROLL No:- 8953

Page No.
Date

Q2 cont.

Final o.p:-

50 40 30 20 10.

i.e. reverse of the queue.

Q3 $((2+3^{*}6)/(9-8/2)+9)^{*}7$

(A) (B) (C)

stack 1:-

*
+
(
(

(A)

o.p:-
23

stack 2:-

/
-
(
/
(

(B)

o.p
236*+982

stack 3:-

*
+
X
(

(C)

o.p:-
236*+982/-19+

o.p:-

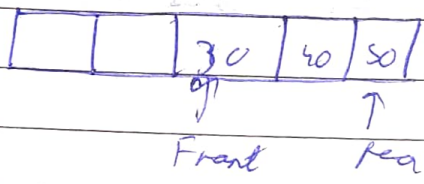
o.p
236*+982/-19+7*

↳ empty

NAME:- BREMDAM LUCAS ROLL No:- 8953

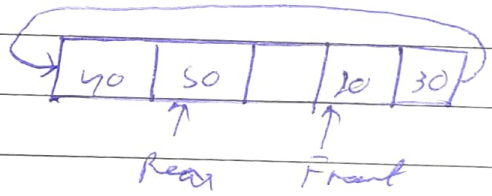
Page No.
Date

Q:4 The limitations of the linear queue is that even if there is empty space in the front of the queue it cannot be utilized

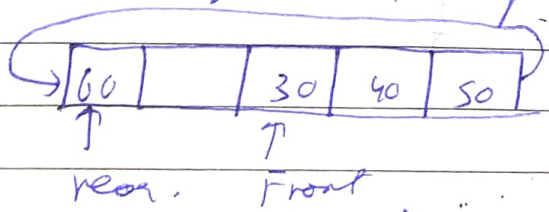


This limitations was overcome using a circular queue.
This was done using the mod operator (%)
The expression is

i = i % (Len of Array).



(b) State of the circular queue after the operations
enqueue(10), enqueue(20), enqueue(30), enqueue(40) enqueue(50), dequeue(), dequeue(), enqueue(60)



NAME:- BREMDAM LUCAS ROLL NO:- 8953

Page No.
Date

Q5 Write the iterative and recursive code to count number of nodes in singly LL.

(i) Iterative method.

```
int count_LL (LL *start)
{
    node *p;
    p = start;
    int i = 0;
    while (p != NULL)
    {
        i++;
        p = p -> next;
    }
    return i;
}
```

(ii) Recursive Method:-

```
int count_LL (node *p)
{
    if (p == NULL)
    { return 0; }
    else
    {
        return (1 + count_LL (p -> next));
    }
}
```

NAME:- BRENDAN LUCAS ROLL No:- 8953

Page No.
Date

q6 Write following methods that operate on stack using Singly Linked List:

(i)
```
push(ele, Stack *head)
{
  node *p;
  p = (node *)malloc(sizeof(node));
  p->data = ele;
  p->next = head;
  head = p;
}
```

(ii)
```
pop(stack *head)          // On next page.
{
  node *p;
  p = head;
  head = p->next;
  return p->data
}
```

(iii)
```
isEmpty(Stack *start)
{
  return start == NULL;
}
```

(iv)
```
peek(stack *head)
{
  if(isEmpty(head))
  { printf("Stack is empty"); }
  else
  { return head->data; }
}
```

NAME:- BRENDAN LUCAS ROLL No:- 8953

Page No.
Date

(iii)
```
pop ( stack * head)
{
  if ( isEmpty (head))
  {
    printf ("Stack is Empty / Stack Underflow");
  }
  else
  {
    node * p;
    p = head;
    head = p -> next;
    return p -> data;
  }
}
```