|  |  |  |
| --- | --- | --- |
|  | Name: Md Habibur Rony  Student ID: 984582  Weekday: Week 1- Day 2 |  |

Answer to the Q. No. R-2.1:

Algorithm InsertBefore(p,e)

newNode<-createNewNode(e)

tmp<-p.prev

tmp.next <- newNode

newNode.next <- p

newNode.prev <- tmp

p.prev <- newNode

Algorithm InsertFirst(e)

newNode<-createNewNode(e)

tmp<-head.next

head.next<-newNode

newNode.next<-tmp

tmp.prev <- newNode

newNode.prev <- head

Algorithm InsertLast(e)

newNode<-createNewNode(e)

tmp<-tail.prev

tail.prev <-newNode

newNode.prev<-tmp

tmp.next <- newNode

newNode.next <- tail

Answer to the Q. No. C-2.1:

|  |  |
| --- | --- |
| Algorithm FindMiddle(list)  Input : list with odd number of nodes  Output : middle position of list  p<- list.first()  q<- list.last()  while p != q do  p<-L.after(p)  q<- list.before(q)  return p | O(1)  O(1)  O(n)  O(n)  O(n)  O(1)  Running Time T(n) = O(n) |

Answer to the Q. No. C-2.2:

|  |  |
| --- | --- |
| S1<-Empty Stack  S2<-Empty Stack  Algorithm Enqueue(val)  If size() = n - 1 Then  throw FullQueueException  S1.push(val) | O(1)  O(1)  O(1)  Running Time T(n) = O(1) |
| Algorithm Dequeue()  If S2.isEmpty() tThen  While !S1.isEmpty() Do  S2.push(S1.pop())  If !S2.isEmpty() Then  return S2.pop()  Else  throw EmptyStackException | O(1)  O(n)  O(n)  O(1)  O(1)  O(1)  Running Time T(n) = O(n) |

Answer to the Q. No. C-2.3:

# Algorithm

q1<-Empty Queue

q2<-Empty Queue

|  |  |
| --- | --- |
| Algorithm Push(val)  If size() = n - 1 Then  throw FullStackException  q1.enqueue(val) | O(1)  O(1)  O(1)  Running Time T(n) = O(1) |
| Algorithm Pop()  If q2.isEmpty() Then  while !q1.isEmpty()  q2.enqueue(q1.dequeue())  If !q2.isEmpty() Then  q2.dequeue()  Else  throw EmptyQueueException | O(1)  O(n)  O(n)  O(1)  O(1)  O(1)  Running Time T(n) = O(n) |

Answer to the Q. No. C-2.4:

**Algorithm Permutation(arry, start, end)**

Input: Number of array arry, start and end is position of array

Output: array which contain all the permutation of array

|  |  |
| --- | --- |
| If start = end Then  print arry  Else  for j<-- start to end do  Swap(arry[start], arry[j])  Permutation(arry, start +1, end)  Swap(arry[start], arry[j])  **Algorithm** Swap **(**a, b)  temp <-- a  a<-- b  b<--temp | O(1)  O(1)  O(1)  O(n)  O(n)  O(n!)  O(n)  O(1)  O(1)  O(1)  Running time T(n) = O(n!) |

Answer to the Q. No. C-2.5:

**Algorithm InsertAtRankZero(obj)**

Input: An object for insertion.

If v.size() = n-1 Then

throw fullException

f<--(f-1+n) Mod n

v[f]<--obj

**Algorithm RemoveAtRankZero()**

If v.IsEmpty() Then

throw EmptyException

f<--(f+1) Mod n

**Algorithm InsertAtRankEnd(obj)**

Input: An object for insertion.

If v.size() = n-1 Then

throw fullException

v[r]<--obj

r<--(f+1) Mod n

**Algorithm RemoveAtRankEnd()**

If v.IsEmpty() Then

throw EmptyException

r<--(r-1+n) Mod n

**Algorithm ElemAtRank(rank)**

Input: A **rank** for getting an element.

Output: An object.

f<--(f + rank) Mod n

return v[f]