Jawahar Education Societys Annasaheb Chudaman Patil College of Engineering, Kharghar, Navi Mumbai

NAME: PRIYUSH BHIMRAO KHOBRAGADE

PRN NO: 211112018

SUBJECT: DATA STRUCTURE LAB

AIM: Implement Linear Queue ADT using Linked List

PAGE NO.:
DATE.: / / 20

Practical No - 07

Aim :- Implement Linear Queue ADT using Linked List.

Theory :-

Simple or linear queue

" Linear queue. on insertion take place from one end while the deletion occur from other end. The end at which the insertion take place is know as the rear end, & the end the deletion takes place is know as front end."

It strictly follow the FIFO rule.

Operation

1) Insert
2) Delect
3) Display

Insert Algorithm

- Step1 : Allocate the space for the new node
- step2 : SET PTR → DATA = VAL
- Step3 : IF FRONT = NULL
  SET FRONT = REAR = PTR
  SET FRONT → NEXT = REAR → NEXT = NULL
  ELSE
  SET REAR → NEXT = PTR
  SET REAR = PTR
  SET REAR → NEXT = NULL
  [END OF IF]
- Step 04 END

Teachers Signature

7.1

AIM: Implement Linear Queue ADT using Linked List

PAGE NO.:
DATE.: / / 20

*Deletion Algorithm

Step 1 : IF FRONT = NULL
write "underflow"
Go to step 5
[END OF IF]
Step 2 : SET PTR = FRONT
Step 3 : SET FRONT = FRONT → NEXT
Step 4 : FREE PTR
Step 5 : END.

Conclusion :-

Hence we implement linear queue ADT using Linked list and perform it's operation.

7.2

Teachers Signature

AIM: Implement Linear Queue ADT using Linked List

Input:

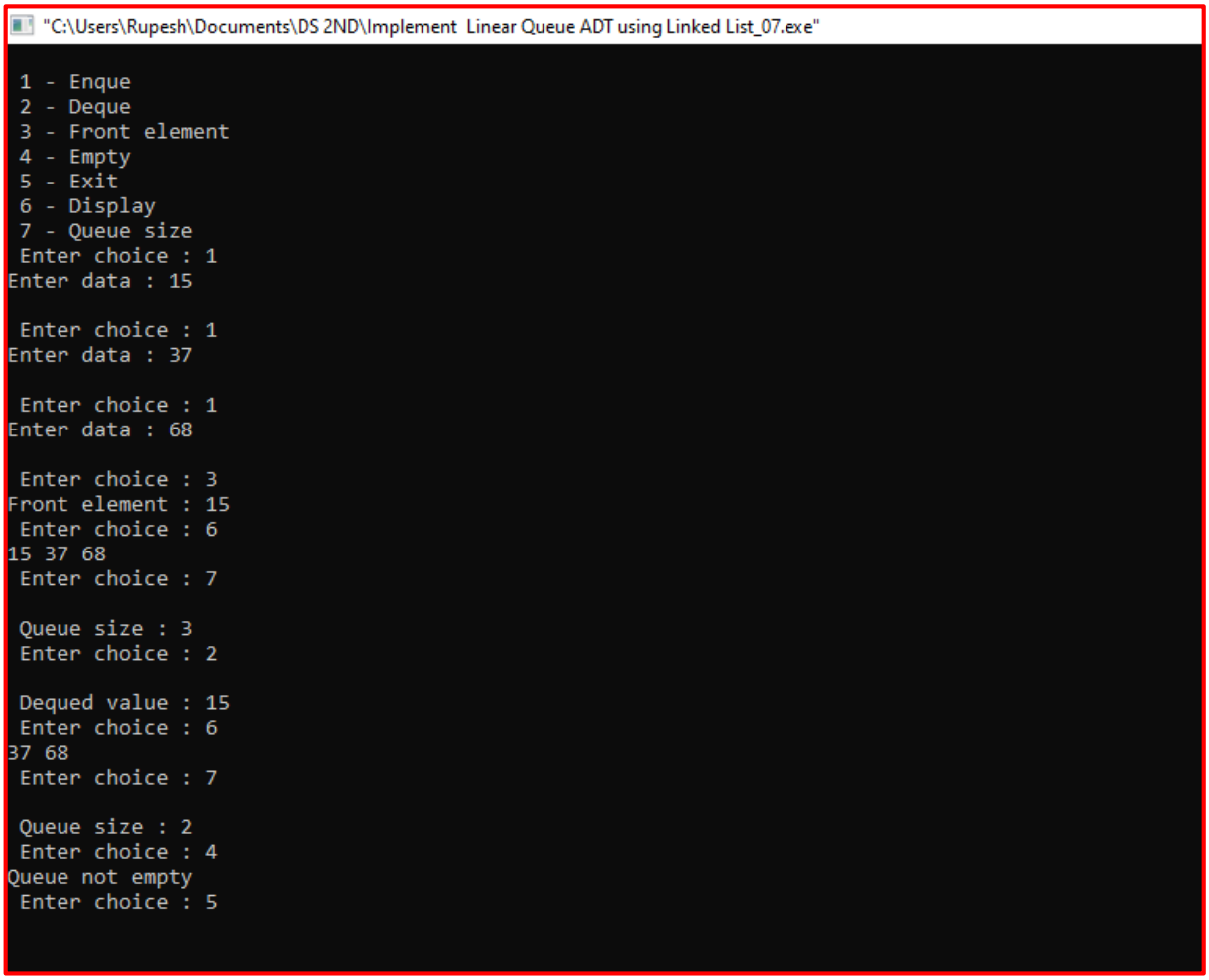
```
/*
 * C Program to Implement Queue Data Structure using Linked List
 */
#include <stdio.h>
#include <stdlib.h>

struct node
{
   int info;
   struct node *ptr;
}*front,*rear,*temp,*front1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

void main()
{
   int no, ch, e;

   printf("\n 1 - Enque");
   printf("\n 2 - Deque");
   printf("\n 3 - Front element");
   printf("\n 4 - Empty");
   printf("\n 5 - Exit");
   printf("\n 6 - Display");
   printf("\n 7 - Queue size");
   create();
   while (1)
   {
      printf("\n Enter choice : ");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
         printf("Enter data : ");
         scanf("%d", &no);
         enq(no);
         break;
      case 2:
         deq();
         break;
      case 3:
         e = frontelement();
         if (e != 0)
            printf("Front element : %d", e);
         else
            printf("\n No front element in Queue as queue is empty");
         break;
      case 4:
         empty();
         break;
      case 5:
         exit(0);
      case 6:
         display();
         break;
      case 7:
         queuesize();
         break;
      default:
         printf("Wrong choice, Please enter correct choice ");
         break;
      }
```

```
71     }
72  }
73
74  /* Create an empty queue*/
75  void create()
76  {
77     front = rear = NULL;
78  }
79
80  /* Returns queue size*/
81  void queuesize()
82  {
83     printf("\n Queue size : %d", count);
84  }
85
86  /* Enqueing the queue*/
87  void enq(int data)
88  {
89     if (rear == NULL)
90     {
91        rear = (struct node*)malloc(1*sizeof(struct node));
92        rear->ptr = NULL;
93        rear->info = data;
94        front = rear;
95     }
96     else
97     {
98        temp=(struct node*)malloc(1*sizeof(struct node));
99        rear->ptr = temp;
100       temp->info = data;
101       temp->ptr = NULL;
102
103       rear = temp;
104    }
105    count++;
106 }
107
108 /* Displaying the queue elements*/
109 void display()
110 {
111    front1 = front;
112
113    if ((front1 == NULL) && (rear == NULL))
114    {
115       printf("Queue is empty");
116       return;
117    }
118    while(front1 != rear)
119    {
120       printf("%d ", front1->info);
121       front1 = front1->ptr;
122    }
123    if (front1 == rear)
124       printf("%d", front1->info);
125 }
126
127 /* Dequeing the queue*/
128 void deq()
129 {
130    front1 = front;
131
132    if (front1 == NULL)
133    {
134       printf("\n Error: Trying to display elements from empty queue");
135       return;
136    }
137    else
138       if (front1->ptr != NULL)
139       {
140          front1 = front1->ptr;
141          printf("\n Dequed value : %d", front->info);
142          free(front);
143          front = front1;
144       }
145       else
```

```
146       {
147          printf("\n Dequed value : %d", front->info);
148          free(front);
149          front = NULL;
150          rear = NULL;
151       }
152       count--;
153 }
154
155 /* Returns the front element of queue */
156 int frontelement()
157 {
158    if ((front != NULL) && (rear != NULL))
159       return(front->info);
160    else
161       return 0;
162 }
163
164 /* Display if queue is empty or not */
165 void empty()
166 {
167     if ((front == NULL) && (rear == NULL))
168       printf("\n Queue empty");
169    else
170      printf("Queue not empty");
171 }
```

AIM: Implement Linear Queue ADT using Linked List

Output:-



Conclusion: -Hence we can implement liner queue ADT using linked list and perform its operation.