```cpp
#include <iostream>
using  namespace  std  ;

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

typedef  struct elemenet Element  ;
typedef  Element *       Queue    ;

Queue  create_queue()
{
    return  NULL  ;
}

bool  empty_queue(Queue que)
{
    if(que == NULL)
        return  true  ;
    else
        return  false ;
}

int  deapth_queue(Queue que)
{
    Queue  ptr    =  que  ;
    int    count  =  0    ;

    while(ptr != NULL)
    {
        count++  ;
        ptr  =  ptr->next  ;
    }

    return  count  ;
}

Queue  enqueue(Queue que, int val)
{
    Queue  elm  =  new  Element  ;
    elm->data   =  val   ;
    elm->next   =  NULL  ;

    if(empty_queue(que))
    {
        que  =  elm  ;
        return  que  ;
    }

    Queue  ptr  =  que  ;

    while(ptr->next != NULL)
        ptr  =  ptr->next  ;

    ptr->next  =  elm  ;

    return  que  ;
}

Queue  dequeue(Queue que, int &val)
{
    if(!empty_queue(que))
    {
        Queue  ptr  =  que  ;

        val  =  que->data  ;
        que  =  que->next  ;
        delete  ptr        ;

        return  que  ;
    }
    else
    {
        cout  <<  "Error, the queue is already empty."  <<  endl  ;
        return  NULL  ;
```

```
 78      }
 79  }
 80
 81  int  front_queue(Queue que)
 82  {
 83      if(empty_queue(que))
 84      {
 85          cout  <<  "Error, Queue is empty."  <<  endl  ;
 86          return  0  ;
 87      }
 88
 89      return  que->data  ;
 90  }
 91
 92  int  rear_queue(Queue que)
 93  {
 94      if(empty_queue(que))
 95      {
 96          cout  <<  "Error, Queue is empty."  <<  endl  ;
 97          return  0  ;
 98      }
100      Queue  ptr  =  que  ;
101
102      while(ptr->next != NULL)
103          ptr  =  ptr->next  ;
104
105      return  ptr->data  ;
106  }
107
108  void  display_queue(Queue que)
109  {
110      Queue  ptr    =  que  ;
111
112      if(empty_queue(que))
113      {
114          cout  <<  "Queue is empty."  <<  endl  ;
115          return  ;
116      }
117
118      while(ptr != NULL)
119      {
120          cout  <<  ptr->data  <<  " "  ;
121          ptr  =  ptr->next  ;
122      }
123      cout  <<  endl  ;
124
125      return  ;
126  }
127
128  int  main()
129  {
130      Queue  que  =  create_queue()  ;
131
134      que  =  enqueue(que, 1)  ;
135      que  =  enqueue(que, 2)  ;
136      que  =  enqueue(que, 3)  ;
137
138      display_queue(que)  ;
139
140      cout  <<  "The front of the queue is  : "  <<  front_queue(que)  <<  endl  ;
142      cout  <<  "The rear of the queue is  : "   <<  rear_queue(que)   <<  endl  ;
145      cout  <<  "The depth of the queue is  : "  <<  deapth_queue(que)  <<  endl  ;
146
147      int  val  ;
148
149      que  =  dequeue(que, val)  ;
150      cout  <<  "The value dequeued from the queue is  : "  <<  val  <<  endl  ;
152
153      que  =  dequeue(que, val)  ;
154      cout  <<  "The value dequeued from the queue is  : "  <<  val  <<  endl  ;
156
157      que  =  dequeue(que, val)  ;
158      cout  <<  "The value dequeued from the queue is  : "  <<  val  <<  endl  ;
159      display_queue(que)  ;
160
161      return  0  ;
162  }
```