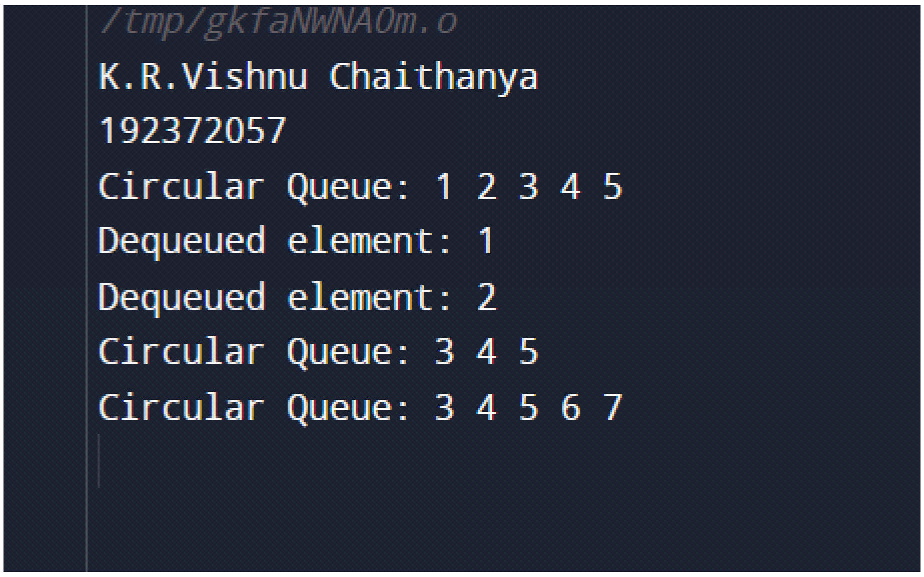
10 March 2024 20:25

```
#include <stdio.h>
#include <stdlib.h>

// Structure for a node in the linked list
struct Node {
   int data;
   struct Node* next;
};

// Structure for the circular queue
struct CircularQueue {
   struct Node* front;
   struct Node* rear;
   int capacity;
   int size;
};

// Function to create a new node
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if (newNode == NULL) {
      printf("Memory allocation failed.\n");
      exit(EXIT_FAILURE);
   }
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}

// Function to create a circular queue
struct CircularQueue* createCircularQueue(int capacity) {
   struct CircularQueue* queue = (struct CircularQueue*)malloc(sizeof(struct CircularQueue));
   if (queue == NULL) {
      printf("Memory allocation failed.\n");
      exit(EXIT_FAILURE);
   }
   queue->front = NULL;
   queue->rear = NULL;
   queue->capacity = capacity;
   queue->size = 0;
   return queue;
}

// Function to check if the circular queue is empty
int isEmpty(struct CircularQueue* queue) {
   return (queue->size == 0);
}

// Function to check if the circular queue is full
int isFull(struct CircularQueue* queue) {
   return (queue->size == queue->capacity);
```

```
}

// Function to enqueue an element into the circular queue
void enqueue(struct CircularQueue* queue, int data) {
    if (isFull(queue)) {
        printf("Queue is full. Cannot enqueue.\n");
        return;
    }

    struct Node* newNode = createNode(data);
    if (isEmpty(queue)) {
        queue->front = newNode;
    } else {
        queue->rear->next = newNode;
    }
    queue->rear = newNode;
    queue->size++;
}

// Function to dequeue an element from the circular queue
int dequeue(struct CircularQueue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty. Cannot dequeue.\n");
        exit(EXIT_FAILURE);
    }

    int data = queue->front->data;
    struct Node* temp = queue->front;
    queue->front = queue->front->next;

    if (queue->front == NULL) {
        queue->rear = NULL;
    }

    free(temp);
    queue->size--;
    return data;
}

// Function to display the circular queue
void display(struct CircularQueue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty.\n");
        return;
    }

    struct Node* temp = queue->front;
    printf("Circular Queue: ");
    do {
        printf("%d ", temp->data);
        temp = temp->next;
    } while (temp != NULL && temp != queue->front);
    printf("\n");
}

int main() {
printf("K.R.Vishnu Chaithanya\n");
printf("192372057\n");
struct CircularQueue* queue = createCircularQueue(5);
    enqueue(queue, 1);
    enqueue(queue, 2);
    enqueue(queue, 3);
    enqueue(queue, 4);
    enqueue(queue, 5);
    display(queue);
    printf("Dequeued element: %d\n", dequeue(queue));
    printf("Dequeued element: %d\n", dequeue(queue));
    display(queue);
    enqueue(queue, 6);
    enqueue(queue, 7);
    display(queue);

    return 0;
}
```



```
/tmp/gkfaNWNA0m.o
K.R.Vishnu Chaithanya
192372057
Circular Queue: 1 2 3 4 5
Dequeued element: 1
Dequeued element: 2
Circular Queue: 3 4 5
Circular Queue: 3 4 5 6 7
```