Rajalakshmi Engineering College

Name: MITHUN CHAKRAVARTHY
Email: 241801158@rajalakshmi.edu.in
Roll no: 241801158
Phone: 9360341516
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 18

Section 1 : MCQ

1. After performing this set of operations, what does the final list look to contain?

InsertFront(10);
InsertFront(20);
InsertRear(30);
DeleteFront();
InsertRear(40);
InsertRear(10);
DeleteRear();
InsertRear(15);
display();

Answer

20 30 40 15

Status : Wrong ***Marks : 0/1***

2. What are the applications of dequeue?

Answer

All the mentioned options

Status : Correct ***Marks : 1/1***

3. What will be the output of the following code?

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 5
typedef struct {
    int* arr;
    int front;
    int rear;
    int size;
} Queue;
Queue* createQueue() {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->arr = (int*)malloc(MAX_SIZE * sizeof(int));
    queue->front = -1;
    queue->rear = -1;
    queue->size = 0;
    return queue;
}
int isEmpty(Queue* queue) {
    return (queue->size == 0);
}
int main() {
    Queue* queue = createQueue();
    printf("Is the queue empty? %d", isEmpty(queue));
    return 0;
}
```

Answer

Is the queue empty? 1

Status : Correct ***Marks : 1/1***

4. In what order will they be removed If the elements “A”, “B”, “C” and “D” are placed in a queue and are deleted one at a time

Answer

ABCD

Status : Correct ***Marks : 1/1***

5. What will the output of the following code?

```
#include <stdio.h>
#include <stdlib.h>
typedef struct {
    int* arr;
    int front;
    int rear;
    int size;
} Queue;
Queue* createQueue() {
    Queue* queue = (Queue*)malloc(sizeof(Queue));
    queue->arr = (int*)malloc(5 * sizeof(int));
    queue->front = 0;
    queue->rear = -1;
    queue->size = 0;
    return queue;
}
int main() {
    Queue* queue = createQueue();
    printf("%d", queue->size);
    return 0;
}
```

Answer

0

Status : Correct ***Marks : 1/1***

6. Which of the following properties is associated with a queue?

Answer

First In First Out

Status : Correct ***Marks : 1/1***

7. When new data has to be inserted into a stack or queue, but there is no available space. This is known as

Answer

overflow

Status : Correct ***Marks : 1/1***

8. What is the functionality of the following piece of code?

```
public void function(Object item)
{
   Node temp=new Node(item,trail);
   if(isEmpty())
   {
      head.setNext(temp);
      temp.setNext(trail);
   }
   else
   {
      Node cur=head.getNext();
      while(cur.getNext()!=trail)
      {
         cur=cur.getNext();
      }
      cur.setNext(temp);
   }
   size++;
```

}

Answer

Insert at the rear end of the dequeue

Status : Correct ***Marks : 1/1***

9. Insertion and deletion operation in the queue is known as

Answer

Enqueue and Dequeue

Status : Correct ***Marks : 1/1***

10. Which operations are performed when deleting an element from an array-based queue?

Answer

Dequeue

Status : Correct ***Marks : 1/1***

11. What does the front pointer in a linked list implementation of a queue contain?

Answer

The address of the first element

Status : Correct ***Marks : 1/1***

12. Which of the following can be used to delete an element from the front end of the queue?

Answer

public Object deleteFront() throws emptyDEQException{if(isEmpty())throw new emptyDEQException("Empty");else{Node temp = head.getNext();Node cur = temp.getNext();Object e = temp.getEle();head.setNext(cur);size--;return e;}}

Status : Correct *Marks : 1/1*

13. What will be the output of the following code?

```
#include <stdio.h>
#define MAX_SIZE 5
typedef struct {
    int arr[MAX_SIZE];
    int front;
    int rear;
    int size;
} Queue;

void enqueue(Queue* queue, int data) {
    if (queue->size == MAX_SIZE) {
        return;
    }
    queue->rear = (queue->rear + 1) % MAX_SIZE;
    queue->arr[queue->rear] = data;
    queue->size++;
}
int dequeue(Queue* queue) {
    if (queue->size == 0) {
        return -1;
    }
    int data = queue->arr[queue->front];
    queue->front = (queue->front + 1) % MAX_SIZE;
    queue->size--;
    return data;
}
int main() {
    Queue queue;
    queue.front = 0;
    queue.rear = -1;
    queue.size = 0;
    enqueue(&queue, 1);
    enqueue(&queue, 2);
    enqueue(&queue, 3);
```

```
    printf("%d ", dequeue(&queue));
    printf("%d ", dequeue(&queue));
    enqueue(&queue, 4);
    enqueue(&queue, 5);
    printf("%d ", dequeue(&queue));
    printf("%d ", dequeue(&queue));
    return 0;
}
```

Answer

1 2 3 4

Status : Correct ***Marks : 1/1***

14. In a linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into a non-empty queue?

Answer

Only rear pointer

Status : Correct ***Marks : 1/1***

15. In linked list implementation of a queue, the important condition for a queue to be empty is?

Answer

FRONT is null

Status : Correct ***Marks : 1/1***

16. A normal queue, if implemented using an array of size MAX_SIZE, gets full when

Answer

Rear = MAX_SIZE – 1

Status : Correct ***Marks : 1/1***

17. The essential condition that is checked before insertion in a queue is?

Answer

Overflow

Status : Correct ***Marks : 1/1***

18. The process of accessing data stored in a serial access memory is similar to manipulating data on a

Answer

Stack

Status : Wrong ***Marks : 0/1***

19. Which one of the following is an application of Queue Data Structure?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

20. Front and rear pointers are tracked in the linked list implementation of a queue. Which of these pointers will change during an insertion into the EMPTY queue?

Answer

Both front and rear pointer

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: MITHUN CHAKRAVARTHY
Email: 241801158@rajalakshmi.edu.in
Roll no: 241801158
Phone: 9360341516
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine a bustling coffee shop, where customers are placing their orders for their favorite coffee drinks. The cafe owner Sheeren wants to efficiently manage the queue of coffee orders using a digital system. She needs a program to handle this queue of orders.

You are tasked with creating a program that implements a queue for coffee orders. Each character in the queue represents a customer's coffee order, with 'L' indicating a latte, 'E' indicating an espresso, 'M' indicating a macchiato, 'O' indicating an iced coffee, and 'N' indicating a nabob.

Customers can place orders and enjoy their delicious coffee drinks.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Enqueue the coffee order into the queue. If the choice is 1, the following input is a space-separated character ('L', 'E', 'M', 'O', 'N').

Choice 2: Dequeue a coffee order from the queue.

Choice 3: Display the orders in the queue.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the queue:

If the choice is 1:

1. Insert the given order into the queue and display "Order for [order] is enqueued." where [order] is the coffee order that is inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue more orders."

If the choice is 2:

1. Dequeue a character from the queue and display "Dequeued Order: " followed by the corresponding order that is dequeued.
2. If the queue is empty without any orders, print "No orders in the queue."

If the choice is 3:

1. The output prints "Orders in the queue are: " followed by the space-separated orders present in the queue.
2. If there are no orders in the queue, print "Queue is empty. No orders available."

If the choice is 4:

1. Exit the program and print "Exiting program"

If any other choice is entered, the output prints "Invalid option."

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 1 L
1 E
1 M
1 O
1 N
1 O
3
2
3
4

Output: Order for L is enqueued.
Order for E is enqueued.
Order for M is enqueued.
Order for O is enqueued.
Order for N is enqueued.
Queue is full. Cannot enqueue more orders.
Orders in the queue are: L E M O N
Dequeued Order: L
Orders in the queue are: E M O N
Exiting program

Answer

```
#include <stdio.h>
#define MAX_SIZE 5

char orders[MAX_SIZE];
int front = -1;
int rear = -1;

void initializeQueue() {
    front = -1;
    rear = -1;
}
int enqueue(char order) {
```

```
    if (rear == MAX_SIZE - 1) {
        printf("Queue is full. Cannot enqueue more orders.\n");
        return 0;
    } else {
        if (front == -1) front = 0;
        rear++;
        orders[rear] = order;
        printf("Order for %c is enqueued.\n", order);
        return 1;
    }
}

void dequeue() {
    if (front == -1 || front > rear) {
        printf("No orders in the queue.\n");
    } else {
        printf("Dequeued Order: %c\n", orders[front]);
        front++;
        if (front > rear) {
            initializeQueue();  // Reset queue when it becomes empty
        }
    }
}

void display() {
    if (front == -1 || front > rear) {
        printf("Queue is empty. No orders available.\n");
    } else {
        printf("Orders in the queue are: ");
        for (int i = front; i <= rear; i++) {
            printf("%c ", orders[i]);
        }
        printf("\n");
    }
}

int main() {
    char order;
    int option;
    initializeQueue();
    while (1) {
        if (scanf("%d", &option) != 1) {
            break;
```

```
        }
        switch (option) {
            case 1:
                if (scanf(" %c", &order) != 1) {
                    break;
                }
                if (enqueue(order)) {
                }
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program");
                return 0;
            default:
                printf("Invalid option.\n");
                break;
        }
    }
    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: MITHUN CHAKRAVARTHY
Email: 241801158@rajalakshmi.edu.in
Roll no: 241801158
Phone: 9360341516
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Amar is working on a project where he needs to implement a special type of queue that allows selective dequeuing based on a given multiple. He wants to efficiently manage a queue of integers such that only elements not divisible by a given multiple are retained in the queue after a selective dequeue operation.

Implement a program to assist Amar in managing his selective queue.

Example

Input:

5

10 2 30 4 50

5

Output:

Original Queue: 10 2 30 4 50

Queue after selective dequeue: 2 4

Explanation:

After selective dequeue with a multiple of 5, the elements that are multiples of 5 should be removed. Therefore, only 10, 30, and 50 should be removed from the queue. The updated Queue is 2 4.

Input Format

The first line contains an integer n, representing the number of elements initially present in the queue.

The second line contains n space-separated integers, representing the elements of the queue.

The third line contains an integer multiple, representing the divisor for selective dequeue operation.

Output Format

The first line of output prints "Original Queue: " followed by the space-separated elements in the queue before the dequeue operation.

The second line prints "Queue after selective dequeue: " followed by the remaining space-separated elements in the queue, after deleting elements that are the multiples of the specified number.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
10 2 30 4 50
5
Output: Original Queue: 10 2 30 4 50
Queue after selective dequeue: 2 4

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a node
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Enqueue function
void enqueue(Node** front, Node** rear, int data) {
    Node* newNode = createNode(data);
    if (*rear == NULL) {
        *front = *rear = newNode;
    } else {
        (*rear)->next = newNode;
        *rear = newNode;
    }
}

// Function to print the queue
void printQueue(Node* front) {
    Node* temp = front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

// Function to selectively dequeue based on a multiple
void selectiveDequeue(Node** front, int multiple) {
    Node *curr = *front, *prev = NULL;
```

```
    while (curr != NULL) {
        if (curr->data % multiple == 0) {
            if (prev == NULL) {
                *front = curr->next;
                free(curr);
                curr = *front;
            } else {
                prev->next = curr->next;
                free(curr);
                curr = prev->next;
            }
        } else {
            prev = curr;
            curr = curr->next;
        }
    }
}

int main() {
    int n, i, val, multiple;
    Node *front = NULL, *rear = NULL;

    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &val);
        enqueue(&front, &rear, val);
    }

    scanf("%d", &multiple);

    printf("Original Queue: ");
    printQueue(front);
    printf("\n");

    selectiveDequeue(&front, multiple);

    printf("Queue after selective dequeue: ");
    printQueue(front);
    printf("\n");

    return 0;
```

}

Status : Correct ***Marks : 10/10***

2. Problem Statement

You are tasked with developing a simple ticket management system for a customer support department. In this system, customers submit support tickets, which are processed in a First-In-First-Out (FIFO) order. The system needs to handle the following operations:

Ticket Submission (Enqueue Operation): New tickets are submitted by customers. Each ticket is assigned a unique identifier (represented by an integer). When a new ticket arrives, it should be added to the end of the queue.

Ticket Processing (Dequeue Operation): The support team processes tickets in the order they are received. The ticket at the front of the queue is processed first. After processing, the ticket is removed from the queue.

Display Ticket Queue: The system should be able to display the current state of the ticket queue, showing the sequence of ticket identifiers from front to rear.

Input Format

The first input line contains an integer n, the number of tickets submitted by customers.

The second line consists of a single integer, representing the unique identifier of each submitted ticket, separated by a space.

Output Format

The first line displays the "Queue: " followed by the ticket identifiers in the queue after all tickets have been submitted.

The second line displays the "Queue After Dequeue: " followed by the ticket identifiers in the queue after processing (removing) the ticket at the front.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 6
14 52 63 95 68 49
Output: Queue: 14 52 63 95 68 49
Queue After Dequeue: 52 63 95 68 49

Answer

```
#include <stdio.h>

#define MAX 20

int main() {
    int queue[MAX];
    int front = 0, rear = 0;
    int n, i;

    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        scanf("%d", &queue[rear]);
        rear++;
    }

    printf("Queue: ");
    for (i = front; i < rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");

    front++;

    printf("Queue After Dequeue: ");
    for (i = front; i < rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
```

```
    return 0;
}
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Guide Harish in developing a simple queue system for a customer service center. The customer service center can handle up to 25 customers at a time. The queue needs to support basic operations such as adding a customer to the queue, serving a customer (removing them from the queue), and displaying the current queue of customers.

Use an array for implementation.

Input Format

The first line of the input consists of an integer N, the number of customers arriving at the service center.

The second line consists of N space-separated integers, representing the customer IDs in the order they arrive.

Output Format

After serving the first customer in the queue, display the remaining customers in the queue.

If a dequeue operation is attempted on an empty queue, display "Underflow".

If the queue is empty, display "Queue is empty".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
101 102 103 104 105
Output: 102 103 104 105

Answer

```
#include <stdio.h>

#define MAX 25

int main() {
    int queue[MAX];
    int front = 0, rear = 0;
    int n, i, id;

    scanf("%d", &n);

    if (n == 0) {
        scanf("%d", &id);
        printf("Underflow\n");
        printf("Queue is empty\n");
        return 0;
    }

    for (i = 0; i < n; i++) {
        scanf("%d", &queue[rear]);
        rear++;
    }

    front++;

    if (front == rear) {
        printf("Queue is empty\n");
    } else {
        for (i = front; i < rear; i++) {
            printf("%d ", queue[i]);
```

```
    }
    printf("\n");
  }

  return 0;
}
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Sharon is developing a queue using an array. She wants to provide the functionality to find the Kth largest element. The queue should support the addition and retrieval of the Kth largest element effectively. The maximum capacity of the queue is 10.

Assist her in the program.

Input Format

The first line of input consists of an integer N, representing the number of elements in the queue.

The second line consists of N space-separated integers.

The third line consists of an integer K.

Output Format

For each enqueued element, print a message: "Enqueued: " followed by the element.

The last line prints "The [K]th largest element: " followed by the Kth largest element.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 45 93 87 25
4

Output: Enqueued: 23
Enqueued: 45
Enqueued: 93
Enqueued: 87
Enqueued: 25
The 4th largest element: 25

Answer

```
// You are using GCC
#include <stdio.h>

#define MAX 10

void sortDescending(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            if (arr[i] < arr[j]) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

int main() {
    int queue[MAX];
    int n, k, i;

    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &queue[i]);
        printf("Enqueued: %d\n", queue[i]);
    }
```

```
    scanf("%d", &k);

    sortDescending(queue, n);

    printf("The %dth largest element: %d\n", k, queue[k - 1]);

    return 0;
}
```

Status : Correct **Marks : 10/10**

5. Problem Statement

You've been assigned the challenge of developing a queue data structure using a linked list.

The program should allow users to interact with the queue by enqueuing positive integers and subsequently dequeuing and displaying elements.

Input Format

The input consists of a series of integers, one per line. Enter positive integers into the queue.

Enter -1 to terminate input.

Output Format

The output prints the space-separated dequeued elements.

Refer to the sample output for the exact text and format.

Sample Test Case

Input: 1
2
3
4

-1

Output: Dequeued elements: 1 2 3 4

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

struct Node* front = NULL;
struct Node* rear = NULL;

void enqueue(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (rear == NULL) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
}

void dequeueAndPrint() {
    struct Node* temp;
    printf("Dequeued elements: ");
    while (front != NULL) {
        printf("%d ", front->data);
        temp = front;
        front = front->next;
        free(temp);
    }
    printf("\n");
    rear = NULL;
}
```

```
int main() {
    int input;
    while (1) {
        scanf("%d", &input);
        if (input == -1)
            break;
        if (input >= 0)
            enqueue(input);
    }

    dequeueAndPrint();
    return 0;
}
```

Status : Correct *Marks : 10/10*