Rajalakshmi Engineering College

Name: KEERTHI PRIYA T
Email: 240701258@rajalakshmi.edu.in
Roll no: 2116240701258
Phone: 7397397221
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

A customer support system is designed to handle incoming requests using a queue. Implement a linked list-based queue where each request is represented by an integer. After processing the requests, remove any duplicate requests to ensure that each request is unique and print the remaining requests.

Input Format

The first line of input consists of an integer N, representing the number of requests to be enqueued.

The second line consists of N space-separated integers, each representing a request.

Output Format

The output prints space-separated integers after removing the duplicate requests.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
2 4 2 7 5

Output: 2 4 7 5

Answer

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

#define MAX_REQUEST 101
struct Node {
   int data;
   struct Node* next;
};

struct Queue {
   struct Node *front, *rear;
};

struct Node* newNode(int data) {
   struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
   temp->data = data;
   temp->next = NULL;
   return temp;
}

void initQueue(struct Queue* q) {
   q->front = q->rear = NULL;
}

void enqueue(struct Queue* q, int data) {
   struct Node* temp = newNode(data);
   if (q->rear == NULL) {
       q->front = q->rear = temp;
       return;
   }
   q->rear->next = temp;
   q->rear = temp;
}

void printUnique(struct Queue* q) {
   int seen[MAX_REQUEST] = {0};
   struct Node* current = q->front;

   while (current != NULL) {
       if (!seen[current->data]) {
           printf("%d ", current->data);
           seen[current->data] = 1;
       }
       current = current->next;
   }
   printf("\n");
}

int main() {
   int n;
   scanf("%d", &n);

   struct Queue q;
   initQueue(&q);

   for (int i = 0; i < n; ++i) {
       int req;
       scanf("%d", &req);
       enqueue(&q, req);
   }

   printUnique(&q);
   return 0;
}
```

Status : Correct *Marks : 10/10*

2. Problem Statement

Saran is developing a simulation for a theme park where people wait in a queue for a popular ride.

Each person has a unique ticket number, and he needs to manage the queue using a linked list implementation.

Your task is to write a program for Saran that reads the number of people in the queue and their respective ticket numbers, enqueue them, and then calculate the sum of all ticket numbers to determine the total ticket value present in the queue.

Input Format

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

The second line consists of N space-separated integers, representing the ticket numbers.

Output Format

The output prints an integer representing the sum of all ticket numbers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
2 4 6 7 5
Output: 24

Answer

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

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

```
struct Queue {
   struct Node *front, *rear;
};

struct Node* newNode(int data) {
   struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
   temp->data = data;
   temp->next = NULL;
   return temp;
}

void initQueue(struct Queue* q) {
   q->front = q->rear = NULL;
}

void enqueue(struct Queue* q, int data) {
   struct Node* temp = newNode(data);
   if (q->rear == NULL) {
      q->front = q->rear = temp;
      return;
   }
   q->rear->next = temp;
   q->rear = temp;
}

int sumQueue(struct Queue* q) {
   int sum = 0;
   struct Node* current = q->front;
   while (current != NULL) {
      sum += current->data;
      current = current->next;
   }
   return sum;
}

int main() {
   int n;
   scanf("%d", &n);

   struct Queue q;
   initQueue(&q);
```

```
    for (int i = 0; i < n; ++i) {
        int ticket;
        scanf("%d", &ticket);
        enqueue(&q, ticket);
    }

    int total = sumQueue(&q);
    printf("%d\n", total);

    return 0;
}
```

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

3. Problem Statement

Imagine you are developing a basic task management system for a small team of software developers. Each task is represented by an integer, where positive integers indicate valid tasks and negative integers indicate erroneous tasks that need to be removed from the queue before processing.

Write a program using the queue with a linked list that allows the team to add tasks to the queue, remove all erroneous tasks (negative integers), and then display the valid tasks that remain in the queue.

Input Format

The first line consists of an integer N, representing the number of tasks to be added to the queue.

The second line consists of N space-separated integers, representing the tasks. Tasks can be both positive (valid) and negative (erroneous).

Output Format

The output displays the following format:

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

The last line displays the "Queue Elements after Dequeue: " followed by removing all erroneous (negative) tasks and printing the valid tasks remaining in the queue in the order they were enqueued.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
12 -54 68 -79 53

Output: Enqueued: 12
Enqueued: -54
Enqueued: 68
Enqueued: -79
Enqueued: 53
Queue Elements after Dequeue: 12 68 53

Answer

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

// Node structure for the queue
struct Node {
   int data;
   struct Node* next;
};

// Queue structure
struct Queue {
   struct Node* front;
   struct Node* rear;
};

// Create a new node
struct Node* newNode(int data) {
   struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
   temp->data = data;
   temp->next = NULL;
   return temp;
```

```
}

// Initialize queue
void initQueue(struct Queue* q) {
   q->front = q->rear = NULL;
}

// Enqueue operation
void enqueue(struct Queue* q, int data) {
   struct Node* temp = newNode(data);
   if (q->rear == NULL) {
      q->front = q->rear = temp;
   } else {
      q->rear->next = temp;
      q->rear = temp;
   }
   printf("Enqueued: %d\n", data);
}

// Function to remove negative (erroneous) tasks
void removeNegativeTasks(struct Queue* q) {
   struct Node *current = q->front, *prev = NULL;

   while (current != NULL) {
      if (current->data < 0) {
         // Remove node
         if (current == q->front) {
            q->front = current->next;
            if (current == q->rear) // Only one node
               q->rear = NULL;
            free(current);
            current = q->front;
         } else {
            prev->next = current->next;
            if (current == q->rear)
               q->rear = prev;
            free(current);
            current = prev->next;
         }
      } else {
         prev = current;
         current = current->next;
```

```
        }
    }
}

// Print valid queue elements
void printQueue(struct Queue* q) {
    printf("Queue Elements after Dequeue: ");
    struct Node* temp = q->front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Main function
int main() {
    int n;
    scanf("%d", &n);

    struct Queue q;
    initQueue(&q);

    for (int i = 0; i < n; i++) {
        int task;
        scanf("%d", &task);
        enqueue(&q, task);
    }

    removeNegativeTasks(&q);
    printQueue(&q);

    return 0;
}
```

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