Rajalakshmi Engineering College

Name: Parameswari P
Email: 240701378@rajalakshmi.edu.in
Roll no: 240701378
Phone: 9500133836
Branch: REC
Department: I CSE FD
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>
typedef struct Node {
   int data;
   struct Node *next;
} Node;

Node* createNode(int data) {
   Node *newNode = (Node*)malloc(sizeof(Node));
   if(!newNode) {
      printf("Memory error\n");
      exit(1);
   }
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}


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

```
}

void removeDuplicates(Node *front) {
   Node *curr = front;
   while(curr != NULL) {
      Node *runner = curr;
      while(runner->next != NULL) {
         if(runner->next->data == curr->data) {
            Node *dup = runner->next;
            runner->next = runner->next->next;
            free(dup);
         } else {
            runner = runner->next;
         }
      }
      curr = curr->next;
   }
}

void printQueue(Node *front) {
   Node *temp = front;
   while(temp != NULL) {
      printf("%d ", temp->data);
      temp = temp->next;
   }
   printf("\n");
}

void freeList(Node *front) {
   Node *temp;
   while(front != NULL) {
      temp = front;
      front = front->next;
      free(temp);
   }
}

int main() {
   int n, data;
   Node *front = NULL, *rear = NULL;

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

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

    removeDuplicates(front);

    printQueue(front);

    freeList(front);
    return 0;
}
```

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

2. Problem Statement

Manoj is learning data structures and practising queues using linked lists. His professor gave him a problem to solve. Manoj started solving the program but could not finish it. So, he is seeking your assistance in solving it.

The problem is as follows: Implement a queue with a function to find the Kth element from the end of the queue.

Help Manoj with 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, representing the queue elements.

The third line consists of an integer K.

Output Format

The output prints an integer representing the Kth element from the end of the queue.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
2 4 6 7 5
3

Output: 6

Answer

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


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


Node* createNode(int data) {
   Node *newNode = (Node*)malloc(sizeof(Node));
   if (!newNode) {
      printf("Memory error\n");
      exit(1);
   }
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}


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

```
    }
}

int kthFromEnd(Node *front, int k) {
    Node *fast = front;
    Node *slow = front;
    for (int i = 0; i < k; i++) {
        if (fast == NULL) {
            return -1;
        }
        fast = fast->next;
    }

    while (fast != NULL) {
        fast = fast->next;
        slow = slow->next;
    }

    return slow->data;
}

void freeList(Node *front) {
    Node *temp;
    while (front != NULL) {
        temp = front;
        front = front->next;
        free(temp);
    }
}

int main() {
    int n, data, k;
    Node *front = NULL, *rear = NULL;

    scanf("%d", &n);

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

    int result = kthFromEnd(front, k);
    if (result != -1) {
        printf("%d\n", result);
    } else {
        printf("Invalid K\n");
    }

    freeList(front);
    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>
typedef struct Node {
   int data;
   struct Node *next;
} Node;

Node* createNode(int data) {
   Node *newNode = (Node*)malloc(sizeof(Node));
   if (!newNode) {
      printf("Memory error\n");
      exit(1);
   }
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}

void enqueue(Node **front, Node **rear, int data) {
   Node *newNode = createNode(data);
   if (*rear == NULL) {
        *front = *rear = newNode;
    } else {
        (*rear)->next = newNode;
        *rear = newNode;
    }
    printf("Enqueued: %d\n", data);
}

void removeNegatives(Node **front, Node **rear) {
    while (*front != NULL && (*front)->data < 0) {
        Node *temp = *front;
        *front = (*front)->next;
        free(temp);
    }
    if (*front == NULL) {
        *rear = NULL;
        return;
    }
    Node *curr = *front;
    while (curr->next != NULL) {
        if (curr->next->data < 0) {
            Node *temp = curr->next;
            curr->next = curr->next->next;
            free(temp);
        } else {
            curr = curr->next;
        }
    }
    *rear = curr;
}

void printQueue(Node *front) {
    Node *temp = front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

void freeList(Node *front) {
    Node *temp;
    while (front != NULL) {
        temp = front;
        front = front->next;
        free(temp);
    }
}

int main() {
    int n, data;
    Node *front = NULL, *rear = NULL;

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

    removeNegatives(&front, &rear);

    printf("Queue Elements after Dequeue: ");
    printQueue(front);

    freeList(front);
    return 0;
}
```

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