Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology

Name :Dumpeta Varshitha . Email :varshithadumpeta@gmail.com
Roll no:23071A1023 Phone :8919552327
Branch :VNRVJIET Department :EIE
Batch :2027 Degree :B.Tech EIE

2023_27_II_EIE 1_Data Structures_lab

DATA STRUCTURES_CODING_WEEK 4

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : CODING

1. Problem Statement

Implement a deque using a doubly linked list and provide functions to add elements at the front and back, remove elements from the front and back, and print the deque accordingly.

Answer

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

// Define the structure for a node in the doubly linked list
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

typedef struct Node Node;
```

```
// Define the structure for the deque
struct Deque {
   Node* front;
   Node* rear;
};

typedef struct Deque Deque;

// Function to initialize an empty deque
void initializeDeque(Deque* dq) {
   dq->front = NULL;
   dq->rear = NULL;
}

// Function to check if the deque is empty
int isEmpty(Deque* dq) {
   return (dq->front == NULL);
}

// Function to insert an element at the front of the deque
void insertFront(Deque* dq, int key) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   if (newNode == NULL) {
      printf("Memory allocation error\n");
      exit(1);
   }
   newNode->data = key;
   newNode->prev = NULL;
   newNode->next = dq->front;

   if (isEmpty(dq)) {
      dq->rear = newNode;
   } else {
      dq->front->prev = newNode;
   }

   dq->front = newNode;
}

// Function to insert an element at the rear of the deque
void insertRear(Deque* dq, int key) {
   Node* newNode = (Node*)malloc(sizeof(Node));
```

```
    if (newNode == NULL) {
        printf("Memory allocation error\n");
        exit(1);
    }
    newNode->data = key;
    newNode->next = NULL;
    newNode->prev = dq->rear;

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

    dq->rear = newNode;
}

// Function to delete an element from the front of the deque
void deleteFront(Deque* dq) {
    if (isEmpty(dq)) {
        printf("Deque is empty\n");
        return;
    }

    Node* temp = dq->front;
    dq->front = dq->front->next;
    free(temp);

    if (dq->front == NULL) {
        dq->rear = NULL;
    } else {
        dq->front->prev = NULL;
    }
}

// Function to delete an element from the rear of the deque
void deleteRear(Deque* dq) {
    if (isEmpty(dq)) {
        printf("Deque is empty\n");
        return;
    }
```

```
    Node* temp = dq->rear;
    dq->rear = dq->rear->prev;
    free(temp);

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

// Function to get the front element of the deque
int getFront(Deque* dq) {
    if (isEmpty(dq)) {
        printf("Deque is empty\n");
        return -1;
    }
    return dq->front->data;
}

// Function to get the rear element of the deque
int getRear(Deque* dq) {
    if (isEmpty(dq)) {
        printf("Deque is empty\n");
        return -1;
    }
    return dq->rear->data;
}

// Function to print the deque from front to rear
void printDeque(Deque* dq) {
    if (isEmpty(dq)) {
        printf("Deque is empty\n");
        return;
    }

    Node* current = dq->front;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}
```

```
int main() {
    Deque dq;
    initializeDeque(&dq);

    int size, choice, value;
    scanf("%d", &size);

    for (int i = 0; i < size; i++) {
        scanf("%d", &choice);
        if (choice == 1) {
            scanf("%d", &value);
            insertFront(&dq, value);
        } else if (choice == 2) {
            scanf("%d", &value);
            insertRear(&dq, value);
        } else {
            printf("Invalid choice\n");
        }
    }

    printf("Original Deque: ");
    printDeque(&dq);

    deleteFront(&dq);
    deleteRear(&dq);
    printf("Deque after removing front and rear elements: ");
    printDeque(&dq);

    int new_front, new_rear;
    scanf("%d", &new_front);
    insertFront(&dq, new_front);
    scanf("%d", &new_rear);
    insertRear(&dq, new_rear);

    printf("Deque after adding new front and rear elements: ");
    printDeque(&dq);

    return 0;
}
```

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

2. Problem Statement

You are required to implement a program that performs operations on a linked list. The program should allow the user to input the number of elements in the linked list, the elements themselves, and a value to search within the list.

The program should output the constructed linked list and indicate whether the specified value is present in the list.

Answer

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

// Link list node
struct Node
{
    int key;
    struct Node* next;
};

/* Given a reference (pointer to pointer) to
the head of a list and an int, push a new
node on the front of the list. */
void push(struct Node** head_ref, int new_key)
{
    // Allocate node
    struct Node* new_node =
        (struct Node*)malloc(sizeof(struct Node));

    // Put in the key
    new_node->key = new_key;

    // Link the old list of the new node
    new_node->next = (*head_ref);

    // Move the head to point to the new node
    (*head_ref) = new_node;
}
```

```
// Checks whether the value x is present
// in linked list
bool search(struct Node* head, int x)
{
    // Initialize current
    struct Node* current = head;
    while (current != NULL)
    {
        if (current->key == x)
            return true;
        current = current->next;
    }
    return false;
}

// Driver code
int main()
{
    // Start with the empty list
    struct Node* head = NULL;
    int n, x;

    // Get the number of elements in the list
    scanf("%d", &n);

    // Get the elements of the list
    for (int i = 0; i < n; ++i)
    {
        int element;
        scanf("%d", &element);
        push(&head, element);
    }

    // Get the value to search
    scanf("%d", &x);

    // Use push() to construct list
    // 14->21->11->30->10
    printf("List: ");
    struct Node* temp = head;
    while (temp != NULL)
```

```
    {
        printf("%d -> ", temp->key);
        temp = temp->next;
    }
    printf("NULL\n");

    search(head, x) ? printf("%d found in the list.\n", x) : printf("%d not found in the
list.\n", x);
    return 0;
}
```

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