DATA STRUCTURE

DAY O2 , 25/07/24 ,CSA0390

1 . write a c programming for linked list singly using all operators.

```
#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 == NULL) {
        perror("Memory allocation failed");
        exit(EXIT_FAILURE);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void insertAtBeginning(Node **head, int data) {
    Node *newNode = createNode(data);
    newNode->next = *head;
    *head = newNode;
}
void insertAtEnd(Node **head, int data) {
    Node *newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node *current = *head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newNode;
    }
}
void deleteNode(Node **head, int key) {
    Node *prev = NULL;
    Node *current = *head;
    while (current != NULL && current->data != key) {
        prev = current;
        current = current->next;
    }
    if (current == NULL) {
        printf("Key %d not found in the list.\n", key);
        return;
    }
    if (prev == NULL) {
        *head = current->next;
    } else {
        prev->next = current->next;
    }
    free(current);
}
void printList(Node *head) {
    Node *current = head;
    printf("Linked List: ");
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}
void freeList(Node *head) {
    Node *current = head;
    Node *next;
    while (current != NULL) {
        next = current->next;
        free(current);
```

```
                current = next;
        }
}
int main()
        Node *head = NULL;
        insertAtEnd(&head, 1);
        insertAtEnd(&head, 2);
        insertAtEnd(&head, 3);
        insertAtBeginning(&head, 0);
        printList(head);
        deleteNode(&head, 3);
        printList(head);
        deleteNode(&head, 0);
        printList(head);
        deleteNode(&head, 5);
        freeList(head);
return 0;
}
```

<u>output</u> :

```
Linked List: 0 -> 1 -> 2 -> 3 -> NULL
Linked List: 0 -> 1 -> 2 -> NULL
Linked List: 1 -> 2 -> NULL
Key 5 not found in the list.
```

2 . write a c programing double n circular using all of operators .

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    int data;
    struct Node *prev;
    struct Node *next;
} Node;
typedef struct DoublyLinkedList {
    Node *head;
    Node *tail;
} DoublyLinkedList;
Node* createNode(int data) {
    Node *newNode = (Node*) malloc(sizeof(Node));
    if (newNode == NULL) {
        perror("Memory allocation failed");
        exit(EXIT_FAILURE);
    }
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}
```

```
doubly linked // Function to initialize a list
DoublyLinkedList* initializeList() {
    DoublyLinkedList *list = (DoublyLinkedList*) malloc(sizeof(DoublyLinkedList));
    if (list == NULL) {
        perror("Memory allocation failed");
        exit(EXIT_FAILURE);
    }
    list->head = NULL;
    list->tail = NULL;
    return list;
}
void insertAtBeginning(DoublyLinkedList *list, int data) {
    Node *newNode = createNode(data);
    if (list->head == NULL) {
        list->head = newNode;
        list->tail = newNode;
    } else {
        newNode->next = list->head;
        list->head->prev = newNode;
        list->head = newNode;
    }
    list->head->prev = list->tail;
    list->tail->next = list->head;
}
```

```
void insertAtEnd(DoublyLinkedList *list, int data) {
    Node *newNode = createNode(data);
    if (list->tail == NULL) {
        list->head = newNode;
        list->tail = newNode;
    } else {
        newNode->prev = list->tail;
        list->tail->next = newNode;
        list->tail = newNode;
    }
    list->tail->next = list->head;
    list->head->prev = list->tail;
}
void printList(DoublyLinkedList *list) {
    if (list->head == NULL) {
        printf("List is empty\n");
        return;
    }
    Node *current = list->head;
    printf("Circular Doubly Linked List: ");
    do {
        printf("%d ", current->data);
        current = current->next;
    } while (current != list->head);
```

```
        printf("\n");
}
void freeList(DoublyLinkedList *list) {
        if (list == NULL) return;
        Node *current = list->head;
        Node *next;
        if (current != NULL) {
                do {
                        next = current->next;
                        free(current);
                        current = next;
                } while (current != list->head);
        }
        free(list);
}
int main() {
        DoublyLinkedList *list = initializeList();
        insertAtEnd(list, 1);
        insertAtEnd(list, 2);
        insertAtEnd(list, 3);
        insertAtBeginning(list, 0);
        printList(list);
      freeList(list);
        return 0;
```

```
}
```

<u>out put</u> :

```
Circular Doubly Linked List: 0 1 2 3
```