# DATA STRUCTURE

## DATE:25/07/24

## 1)Single linked list

```
#include <stdio.h>
#include <stdlib.h>
 struct Node {
   int data;
   struct Node *next;
};
struct Node* createNode(int data) {
   struct Node *newNode = (struct Node*) malloc(sizeof(struct Node));
   if (newNode == NULL) {
      printf("Memory allocation failed.\n");
      return NULL;
   }
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}
 void insertAtBeginning(struct Node **head, int data) {
   struct Node *newNode = createNode(data);
   if (newNode != NULL) {
      newNode->next = *head;
      *head = newNode;
      printf("Inserted %d at the beginning.\n", data);
   }
}
```

```
void insertAtEnd(struct Node **head, int data) {
  struct Node *newNode = createNode(data);
  if (newNode != NULL) {
    if (*head == NULL) {
      *head = newNode;
    } else {
      struct Node *temp = *head;
      while (temp->next != NULL) {
        temp = temp->next;
      }
      temp->next = newNode;
    }
    printf("Appended %d at the end.\n", data);
  }
}
void deleteNode(struct Node **head, int key) {
  struct Node *temp = *head;
  struct Node *prev = NULL;
   if (temp != NULL && temp->data == key) {
    *head = temp->next;
    free(temp);
    printf("Deleted node with data %d.\n", key);
    return;
  }
   while (temp != NULL && temp->data != key) {
    prev = temp;
    temp = temp->next;
  }
```

```
    if (temp == NULL) {
        printf("Node with data %d not found.\n", key);
        return;
    }

        prev->next = temp->next;
    free(temp);
    printf("Deleted node with data %d.\n", key);
}
void traverse(struct Node *head) {
    printf("Linked List: ");
    struct Node *temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
 int main() {
    struct Node *head = NULL;

    // Inserting nodes at the beginning
    insertAtBeginning(&head, 3);
    insertAtBeginning(&head, 7);
    insertAtBeginning(&head, 9);
     insertAtEnd(&head, 11);
    insertAtEnd(&head, 5);
     traverse(head);
```

```
    deleteNode(&head, 7);
   traverse(head);
   return 0;
}
```

**OUTPUT:**

Inserted 3 at the beginning.

Inserted 7 at the beginning.

Inserted 9 at the beginning.

Appended 11 at the end.

Appended 5 at the end.

Linked List: 9 -> 7 -> 3 -> 11 -> 5 -> NULL

Deleted node with data 7.

Linked List: 9 -> 3 -> 11 -> 5 -> NULL

## 2)Double linked list

## Program:

```
#include <stdio.h>
#include <stdlib.h>
 struct Node {
   int data;
   struct Node *prev;
   struct Node *next;
};
 struct Node* createNode(int data) {
   struct Node *newNode = (struct Node*) malloc(sizeof(struct Node));
   if (newNode == NULL) {
```

```
        printf("Memory allocation failed.\n");
        return NULL;
    }
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertAtBeginning(struct Node **head, int data) {
    struct Node *newNode = createNode(data);
    if (newNode != NULL) {
        newNode->next = *head;
        if (*head != NULL) {
            (*head)->prev = newNode;
        }
        *head = newNode;
        printf("Inserted %d at the beginning.\n", data);
    }
}
 void insertAtEnd(struct Node **head, int data) {
    struct Node *newNode = createNode(data);
    if (newNode != NULL) {
        struct Node *last = *head;
        if (*head == NULL) {
            *head = newNode;
        } else {
            while (last->next != NULL) {
```

```
            last = last->next;
        }
        last->next = newNode;
        newNode->prev = last;
    }
    printf("Appended %d at the end.\n", data);
  }
}

void deleteNode(struct Node **head, int key) {
  if (*head == NULL) {
    printf("List is empty. Cannot delete.\n");
    return;
  }
 struct Node *current = *head;
  struct Node *prev = NULL;
  while (current != NULL && current->data != key) {
    prev = current;
    current = current->next;
  }
  if (current == NULL) {
    printf("Node with data %d not found. Cannot delete.\n", key);
    return;
  }
  if (prev != NULL) {
    prev->next = current->next;
  } else {
    *head = current->next; // If deleting the head node
```

```
    }
  if (current->next != NULL) {
        current->next->prev = prev;
    }
     free(current);
    printf("Deleted node with data %d.\n", key);
}
 void traverseForward(struct Node *head) {
    printf("Forward Linked List: ");
    struct Node *temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
 void traverseBackward(struct Node *head) {
    printf("Backward Linked List: ");
    struct Node *temp = head;
    if (temp == NULL) {
        printf("NULL\n");
        return;
    }
    while (temp->next != NULL) {
        temp = temp->next;
    }
    while (temp != NULL) {
        printf("%d -> ", temp->data);
```

```
        temp = temp->prev;
    }
    printf("NULL\n");
}
 int main() {
    struct Node *head = NULL;
     insertAtBeginning(&head, 3);
    insertAtBeginning(&head, 7);
    insertAtBeginning(&head, 9);
     insertAtEnd(&head, 11);
    insertAtEnd(&head, 5);
     traverseForward(head);
    traverseBackward(head);

      deleteNode(&head, 7);
     traverseForward(head);
    traverseBackward(head);
    return 0;
}
```

**OUTPUT:**

```
Inserted 3 at the beginning.
Inserted 7 at the beginning.
Inserted 9 at the beginning.
Appended 11 at the end.
Appended 5 at the end.
Forward Linked List: 9 -> 7 -> 3 -> 11 -> 5 -> NULL
Backward Linked List: 5 -> 11 -> 3 -> 7 -> 9 -> NULL
Deleted node with data 7.
```

Forward Linked List: 9 -> 3 -> 11 -> 5 -> NULL

Backward Linked List: 5 -> 11 -> 3 -> 9 -> NULL

### 3)Circular

### Program:

```
#include <stdio.h>
#include <stdlib.h>
 struct Node {
   int data;
   struct Node *next;
};
 struct Node* createNode(int data) {
   struct Node *newNode = (struct Node*) malloc(sizeof(struct Node));
   if (newNode == NULL) {
      printf("Memory allocation failed.\n");
      return NULL;
   }
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}
 void insertAtBeginning(struct Node **last, int data) {
   struct Node *newNode = createNode(data);
   if (*last == NULL) {
      *last = newNode;
      (*last)->next = *last; // Pointing back to itself
   } else {
      newNode->next = (*last)->next;
      (*last)->next = newNode;
```

```
    }
    printf("Inserted %d at the beginning.\n", data);
}
 void deleteNode(struct Node **last, int key) {
    if (*last == NULL) {
        printf("List is empty. Cannot delete.\n");
        return;
    }
   struct Node *temp = (*last)->next;
    struct Node *prev = *last;
    while (temp != *last && temp->data != key) {
        prev = temp;
        temp = temp->next;
    }
     if (temp == *last && temp->data != key) {
        printf("Node with data %d not found. Cannot delete.\n", key);
        return;
    }
    if (temp == *last) {
        if ((*last)->next == *last) {
            *last = NULL;
        } else {
            *last = prev;
        }
    }
  prev->next = temp->next;
    free(temp);
    printf("Deleted node with data %d.\n", key);
```

```
}
 void traverse(struct Node *last) {
   if (last == NULL) {
      printf("Circular Linked List is empty.\n");
      return;
   }
 struct Node *temp = last->next;
   printf("Circular Linked List: ");
   do {
      printf("%d -> ", temp->data);
      temp = temp->next;
   } while (temp != last->next);
   printf("\n");
}
int main() {
   struct Node *last = NULL;
    insertAtBeginning(&last, 3);
   insertAtBeginning(&last, 7);
   insertAtBeginning(&last, 9);

   traverse(last);
    deleteNode(&last, 7);
    traverse(last);
   return 0;
}
```

**Output:**

Inserted 3 at the beginning.

Inserted 7 at the beginning.

Inserted 9 at the beginning.

Circular Linked List: 9 -> 7 -> 3 ->

Deleted node with data 7.

Circular Linked List: 9 -> 3 ->