Assignment 2

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));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
Void insertEnd(struct Node** head, int data) {
    Struct Node* newNode = createNode(data);
    If (*head == NULL) {
        *head = newNode;
        Return;
    }
    Struct Node* temp = *head;
    While (temp->next != NULL) {
        Temp = temp->next;
    }
    Temp->next = newNode;
}
Void traverseList(struct Node* head) {
```

```
    Struct Node* temp = head;
    While (temp != NULL) {
        Printf("%d -> ", temp->data);
        Temp = temp->next;
    }
    Printf("NULL\n");
}
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);
        Return;
    }
    While (temp != NULL && temp->data != key) {
        Prev = temp;
        Temp = temp->next;
    }
    If (temp == NULL) return;
    Prev->next = temp->next;
    Free(temp);
}
Struct Node* searchNode(struct Node* head, int key) {
    Struct Node* temp = head;
    While (temp != NULL) {
        If (temp->data == key) {
```

```
            Return temp;
        }
        Temp = temp->next;
    }
    Return NULL;
}
Void updateNode(struct Node* head, int oldData, int newData) {
    Struct Node* temp = searchNode(head, oldData);
    If (temp != NULL) {
        Temp->data = newData;
    }
}

Int main() {
    Struct Node* head = NULL;

    insertEnd(&head, 10);
    insertEnd(&head, 20);
    insertEnd(&head, 30);
    printf("Linked List after insertion: ");
    traverseList(head);

    deleteNode(&head, 20);
    printf("Linked List after deletion of 20: ");
    traverseList(head);

    struct Node* foundNode = searchNode(head, 30);
```

```
    if (foundNode != NULL) {
        printf(“Node with data 30 found.\n”);
    } else {
        Printf(“Node with data 30 not found.\n”);
    }

    updateNode(head, 10, 100);
    printf(“Linked List after updating 10 to 100: “);
    traverseList(head);

    return 0;
}
```

Double Linked List:

```
#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));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
```

```
}
Void insertEnd(struct Node** head, int data) {
    Struct Node* newNode = createNode(data);
    If (*head == NULL) {
        *head = newNode;
        Return;
    }
    Struct Node* temp = *head;
    While (temp->next != NULL) {
        Temp = temp->next;
    }
    Temp->next = newNode;
    newNode->prev = temp;
}
Void traverseList(struct Node* head) {
    Struct Node* temp = head;
    While (temp != NULL) {
        Printf("%d -> ", temp->data);
        Temp = temp->next;
    }
    Printf("NULL\n");
}
Void deleteNode(struct Node** head, int key) {
    Struct Node* temp = *head;
    If (temp != NULL && temp->data == key) {
        *head = temp->next;
        If (*head != NULL) {
```

```
            (*head)->prev = NULL;
        }
        Free(temp);
        Return;
    }
    While (temp != NULL && temp->data != key) {
        Temp = temp->next;
    }
    If (temp == NULL) return;
    If (temp->next != NULL) {
        Temp->next->prev = temp->prev;
    }
    If (temp->prev != NULL) {
        Temp->prev->next = temp->next;
    }
    Free(temp);
}
Struct Node* searchNode(struct Node* head, int key) {
    Struct Node* temp = head;
    While (temp != NULL) {
        If (temp->data == key) {
            Return temp;
        }
        Temp = temp->next;
    }
    Return NULL;
}
```

```
// Function to update a node’s data
Void updateNode(struct Node* head, int oldData, int newData) {
    Struct Node* temp = searchNode(head, oldData);
    If (temp != NULL) {
        Temp->data = newData;
    }
}

Int main() {
    Struct Node* head = NULL;

    insertEnd(&head, 10);
    insertEnd(&head, 20);
    insertEnd(&head, 30);
    printf(“Double Linked List after insertion: “);
    traverseList(head);

    deleteNode(&head, 20);
    printf(“Double Linked List after deletion of 20: “);
    traverseList(head);

    struct Node* foundNode = searchNode(head, 30);
    if (foundNode != NULL) {
        printf(“Node with data 30 found.\n”);
    } else {
        Printf(“Node with data 30 not found.\n”);
```

```
    }

    updateNode(head, 10, 100);
    printf("Double Linked List after updating 10 to 100: ");
    traverseList(head);

    return 0;
}
```

Circular 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));
    newNode->data = data;
    newNode->next = newNode; // Point to itself to make it circular
    return newNode;
}
Void insertEnd(struct Node** head, int data) {
    Struct Node* newNode = createNode(data);
    If (*head == NULL) {
        *head = newNode;
        Return;
```

```
    }
    Struct Node* temp = *head;
    While (temp->next != *head) {
        Temp = temp->next;
    }
    Temp->next = newNode;
    newNode->next = *head;
}
Void traverseList(struct Node* head) {
    If (head == NULL) return;
    Struct Node* temp = head;
    Do {
        Printf("%d -> ", temp->data);
        Temp = temp->next;
    } while (temp != head);
    Printf("(head)\n");
}
Void deleteNode(struct Node** head, int key) {
    If (*head == NULL) return;

    Struct Node *temp = *head, *prev = NULL;

    // If the head node itself holds the key to be deleted
    If (temp->data == key && temp->next == *head) {
        *head = NULL;
        Free(temp);
        Return;
```

```
    }

    // If head needs to be removed
    If (temp->data == key) {
        While (temp->next != *head) temp = temp->next;
        Temp->next = (*head)->next;
        Free(*head);
        *head = temp->next;
        Return;
    }

    // Either the node to be deleted is not found
    While (temp->next != *head && temp->data != key) {
        Prev = temp;
        Temp = temp->next;
    }

    // Node to be deleted was found
    If (temp->data == key) {
        Prev->next = temp->next;
        Free(temp);
    }
}

// Function to search for a node with given data
Struct Node* searchNode(struct Node* head, int key) {
    If (head == NULL) return NULL;
```

```
    Struct Node* temp = head;
    Do {
        If (temp->data == key) return temp;
        Temp = temp->next;
    } while (temp != head);
    Return NULL;
}
Void updateNode(struct Node* head, int oldData, int newData) {
    Struct Node* temp = searchNode(head, oldData);
    If (temp != NULL) {
        Temp->data = newData;
    }
}

Int main() {
    Struct Node* head = NULL;

    insertEnd(&head, 10);
    insertEnd(&head, 20);
    insertEnd(&head, 30);
    printf("Circular Linked List after insertion: ");
    traverseList(head);

    deleteNode(&head, 20);
    printf("Circular Linked List after deletion of 20: ");
    traverseList(head);
```

```
    struct Node* foundNode = searchNode(head, 30);
    if (foundNode != NULL) {
        printf(“Node with data 30 found.\n”);
    } else {
        Printf(“Node with data 30 not found.\n”);
    }

    updateNode(head, 10, 100);
    printf(“Circular Linked List after updating 10 to 100: “);
    traverseList(head);

    return 0;
}
```