PROGRAM: LINKED LIST IMPLEMENTATION IN "C"

```
#include <stdio.h>
#include <stdlib.h>

struct node {
   int data;
   struct node *next;
};

struct node *insertAtBeginning(struct node *start, int data) {
   struct node *newnode = (struct node *)malloc(sizeof(struct node));
   if (newnode == NULL) {
      printf("Memory allocation failed.\n");
      return start;
   }

   newnode->data = data;
   newnode->next = start;
   return newnode;
}

struct node *insertAtEnd(struct node *start, int data) {
   struct node *newnode = (struct node *)malloc(sizeof(struct node));
   if (newnode == NULL) {
      printf("Memory allocation failed.\n");
      return start;
   }

   newnode->data = data;
   newnode->next = NULL;

   if (start == NULL) {
      return newnode;
   }
```

```
    struct node *ptr = start;
    while (ptr->next != NULL) {
        ptr = ptr->next;
    }
    ptr->next = newnode;

    return start;
}

struct node *insertBeforeNode(struct node *start, int data, int target) {
    struct node *newnode = (struct node *)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Memory allocation failed.\n");
        return start;
    }

    newnode->data = data;

    if (start == NULL) {
        newnode->next = NULL;
        return newnode;
    }

    if (start->data == target) {
        newnode->next = start;
        return newnode;
    }

    struct node *prev = NULL;
    struct node *current = start;
    while (current != NULL && current->data != target) {
        prev = current;
        current = current->next;
    }
```

```
    if (current == NULL) {
        printf("Target node not found.\n");
        free(newnode);
        return start;
    }

    prev->next = newnode;
    newnode->next = current;

    return start;
}

struct node *insertAfterNode(struct node *start, int data, int target) {
    struct node *newnode = (struct node *)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Memory allocation failed.\n");
        return start;
    }

    newnode->data = data;
    newnode->next = NULL;

    if (start == NULL) {
        return newnode;
    }

    struct node *current = start;
    while (current != NULL && current->data != target) {
        current = current->next;
    }

    if (current == NULL) {
        printf("Target node not found.\n");
        free(newnode);
```

```
        return start;
    }

    newnode->next = current->next;
    current->next = newnode;

    return start;
}

struct node *deleteFirstNode(struct node *start) {
    if (start == NULL) {
        printf("List is empty.\n");
        return start;
    }

    struct node *temp = start;
    start = start->next;
    free(temp);

    return start;
}

struct node *deleteLastNode(struct node *start) {
    if (start == NULL) {
        printf("List is empty.\n");
        return start;
    }

    if (start->next == NULL) {
        free(start);
        return NULL;
    }

    struct node *ptr = start;
    while (ptr->next->next != NULL) {
```

```
        ptr = ptr->next;
    }

    free(ptr->next);
    ptr->next = NULL;

    return start;
}

struct node *deleteAfterNode(struct node *start, int target) {
    if (start == NULL) {
        printf("List is empty.\n");
        return start;
    }

    struct node *current = start;
    while (current != NULL && current->data != target) {
        current = current->next;
    }

    if (current == NULL) {
        printf("Target node not found.\n");
        return start;
    }

    struct node *temp = current->next;
    if (temp != NULL) {
        current->next = temp->next;
        free(temp);
    } else {
        printf("No node to delete after the target node.\n");
    }

    return start;
}
```

```
struct node *deleteBeforeNode(struct node *start, int target) {
    if (start == NULL || start->next == NULL) {
        printf("Cannot delete before the target node.\n");
        return start;
    }

    if (start->next->data == target) {
        struct node *temp = start;
        start = start->next;
        free(temp);
        return start;
    }

    struct node *prev = start;
    struct node *current = start->next;
    while (current->next != NULL && current->next->data != target) {
        prev = current;
        current = current->next;
    }

    if (current->next == NULL) {
        printf("Target node not found.\n");
        return start;
    }

    prev->next = current->next;
    free(current);

    return start;
}

void display(struct node *start) {
    struct node *ptr = start;
    printf("Elements are: ");
```

```
    while (ptr != NULL) {
        printf("%d ", ptr->data);
        ptr = ptr->next;
    }
    printf("\n");
}

void freeLinkedList(struct node *start) {
    struct node *ptr = start;
    while (ptr != NULL) {
        struct node *temp = ptr;
        ptr = ptr->next;
        free(temp);
    }
}

int main() {
    struct node *start = NULL;
    int choice, num, target;

    do {
        printf("\nMenu:\n");
        printf("1. Insert at beginning\n");
        printf("2. Insert at end\n");
        printf("3. Insert before node\n");
        printf("4. Insert after node\n");
        printf("5. Delete first node\n");
        printf("6. Delete last node\n");
        printf("7. Delete after node\n");
        printf("8. Delete before node\n");
        printf("9. Display\n");
        printf("10. Quit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
switch (choice) {
    case 1:
        printf("Enter data: ");
        scanf("%d", &num);
        start = insertAtBeginning(start, num);
        break;

    case 2:
        printf("Enter data: ");
        scanf("%d", &num);
        start = insertAtEnd(start, num);
        break;

    case 3:
        printf("Enter data: ");
        scanf("%d", &num);
        printf("Enter target node data: ");
        scanf("%d", &target);
        start = insertBeforeNode(start, num, target);
        break;

    case 4:
        printf("Enter data: ");
        scanf("%d", &num);
        printf("Enter target node data: ");
        scanf("%d", &target);
        start = insertAfterNode(start, num, target);
        break;

    case 5:
        start = deleteFirstNode(start);
        break;

    case 6:
        start = deleteLastNode(start);
```

```
            break;

        case 7:
            printf("Enter target node data: ");
            scanf("%d", &target);
            start = deleteAfterNode(start, target);
            break;

        case 8:
            printf("Enter target node data: ");
            scanf("%d", &target);
            start = deleteBeforeNode(start, target);
            break;

        case 9:
            display(start);
            break;

        case 10:
            printf("Exiting program.\n");
            break;

        default:
            printf("Invalid choice. Please try again.\n");
      }
   } while (choice != 10);

   // Free the allocated memory
   freeLinkedList(start);

   return 0;
}
```

OUTPUT

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 1

Enter data: 50

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 1

Enter data: 40

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 1

Enter data: 30

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 1

Enter data: 20

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 1

Enter data: 10

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 1

Enter data: 0

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 2

Enter data: 60

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 3

Enter data: 5

Enter target node data: 10

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 4

Enter data: 15

Enter target node data: 10

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 9

Elements are: 0 5 10 15 20 30 40 50 60

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 5

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 6

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 7

Enter target node data: 10

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 8

Enter target node data: 10

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 9

Elements are: 10 20 30 40 50

Menu:

1. Insert at beginning

2. Insert at end

3. Insert before node

4. Insert after node

5. Delete first node

6. Delete last node

7. Delete after node

8. Delete before node

9. Display

10. Quit

Enter your choice: 10

Exiting program.