22AIE1112 Data Structures and Algorithms
Lab Sheet 2
Linked List

Date 4/6/2023 **Roll No: AM.EN.U4AIE22009**

1. Insertion at the beginning

```c
// Function to insert a node at the beginning
void insertbegin() {
    struct node* newnode;
    int data;

    newnode = (struct node*)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Unable to allocate memory.");
        return;
    }

    printf("Enter the value of the new node: ");
    scanf("%d", &data);

    newnode->data = data;
    newnode->link = head;
    head = newnode;

    traversal();
}
```

Output

```
enter the number of nodes 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 5
Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
2
Enter the value of the new node: 12
Value is 12. Address of the next node is 0x555555559ac0
Value is 1. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is 0x555555559b40
Value is 5. Address of the next node is (nil)
```

2. Instertion of new node at the end

```c
// Function to insert a node at the end
void insertend() {
    struct node* newnode, *temp;
    int data;

    newnode = (struct node*)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Unable to allocate memory.");
        return;
    }

    printf("Enter the value of the new node: ");
    scanf("%d", &data);

    newnode->data = data;
    newnode->link = NULL;

    if (head == NULL) {
        head = newnode;
    } else {
        temp = head;
        while (temp->link != NULL) {
            temp = temp->link;
        }
        temp->link = newnode;
    }

    traversal();
}
```

Output

```
enter the number of nodes 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 5
Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
3
Enter the value of the new node: 12
Value is 1. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is 0x555555559b40
Value is 5. Address of the next node is 0x555555559b60
Value is 12. Address of the next node is (nil)
```

3. Insertion at a specified position

```c
// Function to insert a node at a specific position
void insertpos() {
    int position, counter = 1;
    int data;
    struct node* newnode, *temp, *prev;

    newnode = (struct node*)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Unable to allocate memory.");
        return;
    }

    printf("Enter the position where to insert: ");
    scanf("%d", &position);

    printf("Enter the value of the new node: ");
    scanf("%d", &data);

    newnode->data = data;
    newnode->link = NULL;

    temp = head;
    prev = NULL;

    while (temp != NULL && counter < position) {
        prev = temp;
        temp = temp->link;
        counter++;
    }

    if (temp == NULL) {
        printf("Invalid position. Inserting at the end.\n");
        prev->link = newnode;
    } else {
        newnode->link = temp;
        if (prev == NULL) {
            head = newnode;
        } else {
            prev->link = newnode;
        }
    }

    traversal();
}
```

Output

```
enter the number of nodes 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 5
Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
4
Enter the position where to insert: 3
Enter the value of the new node: 12
Value is 1. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559b60
Value is 12. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is 0x555555559b40
Value is 5. Address of the next node is (nil)
The length of the list is 7
```

4. Delete the first node.

```c
// Function to delete the first node
void deletebeg() {
    if (head == NULL) {
        printf("List is empty. Unable to delete.");
        return;
    }

    struct node* temp = head;
    head = head->link;
    free(temp);

    traversal();
}
```

Output

```
enter the number of nodes 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 5
Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
6
Value is 2. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is 0x555555559b40
Value is 5. Address of the next node is (nil)
The length of the list is 5
Value is 2. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is (nil)
The length of the list is 4
```

5.Delete the last node.

```
void deleteend(){
    struct node *temp=head;
    struct node *prev=NULL;
    while(temp->link!=NULL){
        prev=temp;
        temp=temp->link;
    }
    prev->link=NULL;
    free(temp);
    traversal();
}
```

Output

```
enter the number of nodes 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 5
Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
7
Value is 1. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is (nil)
The length of the list is 5
```

6. Delete from a specified position

```c
void deletepos() {
    int position, counter = 1;
    struct node* temp = head;
    struct node* prev = NULL;

    if (temp == NULL) {
        printf("List is empty. Unable to delete.");
        return;
    }

    printf("Enter the position to delete: ");
    scanf("%d", &position);
    while (temp != NULL && counter < position) {
        prev = temp;
        temp = temp->link;
        counter++;
    }

    if (temp == NULL) {
        printf("Invalid position. Unable to delete.\n");
        return;
    }

    if (prev == NULL) {
        head = temp->link;
    } else {
        prev->link = temp->link;
    }

    free(temp);

    traversal();
}
```

Output

```
enter the number of nodes 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 5
Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
8. DELETE FROM SPECIFIED POSITION
8
Enter the position to delete: 3
Value is 1. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is 0x555555559b40
Value is 5. Address of the next node is (nil)
The length of the list is 5
```

7. Find the length of the linked list

```c
// Function to traverse and display the linked list
void traversal() {
    struct node* temp = head;
    int i = 1;

    while (temp != NULL) {
        printf("Value is %d. Address of the next node is %p\n", tem
        temp = temp->link;
        i++;
    }

    printf("The length of the list is %d\n", i);
}
```

Output

```
enter the number of nodes 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 5
Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
8. DELETE FROM SPECIFIED POSITION
1
Value is 1. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is 0x555555559b40
Value is 5. Address of the next node is (nil)
The length of the list is 5
Enter the value of the new node:
```

8. Reverse the linked list

```
// Function to reverse the linked list
void reverse() {
    struct node* prev = NULL;
    struct node* current = head;
    struct node* next = NULL;

    while (current != NULL) {
        next = current->link;
        current->link = prev;
        prev = current;
        current = next;
    }

    head = prev;

    traversal();
}
```

Output

```
enter the number of nodes 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 5
Linked List OPERATION MENU
1. FIND LENGTH OF LINKED LIST
2. INSERT NODE TO BEGINNING
3. INSERT NODE TO THE END
4. INSERT NODE TO A SPECIFIC POINT
5. REVERSE THE LINKED LIST
6. DELETE NODE FROM THE BEGINNING
7. DELETE NODE AT THE END
8. DELETE FROM SPECIFIED POSITION
5
Value is 5. Address of the next node is 0x555555559b20
Value is 4. Address of the next node is 0x555555559b00
Value is 3. Address of the next node is 0x555555559ae0
Value is 2. Address of the next node is 0x555555559ac0
Value is 1. Address of the next node is (nil)
The length of the list is 5
```