22/01/24

SURYA Gold
Date Page

8) WAP, to implement singly linked list a) Create a linked list b) insertion of node at beg, any position, end, Display.

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

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

struct Node * createNode(int value) {
    struct Node * creatNode(int value) {
        struct Node * newNode = (struct Node *)
                malloc(sizeof(struct Node));
    if (newNode == NULL) {
    printf("Memory allocation failed\n");
    }
    newNode → data = value;
    newNode → next = NULL;
    return newNode;
}

struct Node * insertAtBeginning(struct Node
                * head, int value) {
    struct Node * newNode = creatNode(value);
    newNode → next = head;
    return newNode;
}
```

```
void insertAtPosition (struct Node * previous, int value) {
    if (previous == NULL) {
        printf("Previous node cannot be NULL\n");
        return;
    }
    struct Node * newNode = createNode(value);
    newNode -> next = previous -> next;
    previous -> next = newnode;
}

void insertAtEnd (struct Node * head, int value)
{
    struct Node * newNode = createNode(value);
    struct Node * current = head;
    while (current -> next != NULL) {
        current = current -> next;
    }
    current -> next = newNode;
}

void displayList (struct Node * head) {
    struct Node * current = head;
    while (current != NULL) {
        printf("%d -> ", current -> data);
        current = current -> next;
    }
    printf("NULL\n");
}
```

```
int main() {
    struct Node * head = NULL;
    int choice, value, position;
    do{
    printf("1. Insert at beg");
    printf("2. Insert at any position");
    printf("3. Insert at end");
    printf("4. Display");
      printf("5. Exit");
    printf("Enter choice:");
    scanf("%od", &choice);

switch choice{
  case 1: printf("Enter value);
          scanf("%od", &value);
          head = insertAtBeginning(head, value);
          break;
  case 2: printf("Enter the position after which
          to insert:");
          scanf("%od", &position);
          printf("Enter the value to insert at position
                  %od:", position)
          scanf("%od", &value);
          insertAtPosition(head, value);
          break;
  case 3:
          printf("Enter value to insert at end");
          scanf("%od", &value);
          insertAtEnd(head, value);
          break;
  case 4: displayList(head);
          break;
  case 5: printf("Exiting");
          break;
  default: printf("Invalid choice");
```

```
} while (choice != 5);
return 0;
}
```

Output:-

Enter your choice: 1
Enter the value to insert at the beginning: 10
Enter your choice: 1
Enter the value to insert at the beginning: 20
Enter your choice: 2
Enter your choice: 2
Enter the value to insert at the end: 30
Enter your choice: 2
Enter the value to insert at the end: 60
Enter your choice: 6

20 → 10 → 30 → 60 → NULL

WAP to implement singly linked list, create linked list Deletion of first element, specified element, last element in the list. Display the contents

```
void delete_first (struct Node * head)
{
    if (head->next == NULL)
    {
        free(head);
        return;
    }
    head = head->next;
    free(p);
}
```

```
struct Node * deleteAtBeginning (struct Node* head)
    if (head == NULL)
    {
    printf ("List is empty");
    return NULL;
    }
    struct Node * newhead = head->next;
    free(head);
    return newhead;
}
```

```
void deleteAtEnd (struct Node * head){
    if (head == NULL){
    printf ("List is empty");
    return;
    }
```

```
if (head->next == NULL)
{
    free(head);
    return;
}
if(head
struct Node * current = head;
struct Node * previous = NULL;
while (current->next != NULL) {
    previous = current;
    current = current->next;
}
free(current);
previous->next = NULL;
}
```

```
void deletionAtPosition(struct Node *head,
                        int position) {
    if (head == NULL) {
        printf("List is empty");
        return NULL;
    }
    if (position == 0) {
        struct Node *temp = head;
        head = head->next;
        free(temp);
        return head;
    }
    struct Node * current = head;
    struct Node * previous = NULL;
    int count = 0;
```

```
while (current != NULL && count < position)
{
    previous = current;
    current = current->next;
    count++;
}
if (current == NULL){
    printf("Invalid position");
    return head;
}
previous->next = current->next;
free(current);
return head;
}
```

Output:

Considering the same linked list as mentioned in previous output

20 → 10 → 30 → 60 → NULL

```
Enter your choice: 3          // delete at beginning
Enter your choice: 5          }// delete at any position
Enter the position to delete: 2
Enter your choice: 4          // delete at end
Enter your choice: 6          // display.
    10 → NULL
```

3/2/2024