22-1-24

1) Implement C program to perform insertion operation (on all the three position) on singly linked list.
WAP to Implement Singly Linked List with following operations
a) Create a linked list
b) Insertion of a node at first position, at any position and at end of list.
Display the contents of the linkedlist

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

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

void InsertAtBeginning(Node **headref,
                       int newdata);
void InsertAtEnd(Node **headref, int newdata);
void InsertAtPosition(Node **headref,
                      int newdata, int pos);
void PrintList(Node *node);

void InsertAtBeginning(Node **headref,
                       int newdata)
{
    Node *newnode = (Node*) malloc(sizeof
                                   (Node));
```

```
    newnode->data = newdata;
    newnode->next = *headref;
    *headref = newnode;
}

void InsertAtEnd(Node **headref,
                 int newdata)
{
    Node *newnode = (Node*) malloc
                    (sizeof(Node));
    Node *last = *headref;
    newnode->data = newdata;
    newnode->next = NULL;
    if (*headref == NULL)
    {
        *headref = newnode;
        return;
    }
    while (last->next != NULL)
        last = last->next;
    last->next = newnode;
}

void InsertAtPosition(Node **headref,
                      int newdata, int pos)
{
    if (*headref == NULL && pos > 1)
    {
        printf("Cannot insert at the
                specified position \n");
        return;
    }
```

DATE: / /

```
Node *temp = *head_ref;
Node *newNode = (Node*) malloc
                (sizeof(Node));
newNode -> data = newdata;
newNode -> next = NULL;
while (--pos > 0) {
  if (temp == NULL) {
    printf ("Cannot insert at the
            specified position \n");
    return;
  }
  temp = temp -> next;
}
if (temp == NULL) {
  printf("Cannot insert at the
          specified position \n");
  return;
}
newNode -> next = temp -> next;
temp -> next = newNode;
}

void PrintList(Node *node){
    while (node != NULL){
      printf ("%d \n", node -> data);
      node = node -> next;
    }
}

int main() {
  int ch, new, pos;
  Node *head = NULL;
```

```
while (ch != 5) {
    printf("Menu\n");
    printf("1. Insert at beginning\n");
    printf("2. Insert at a specific
                position\n");
    printf("3. Insert at end\n");
    printf("4. Display linked list\n");
    printf("5. Exit\n");
    printf("Enter your choice\n");
    scanf("%d", &ch);
    switch(ch) {
    case 1:
        printf("Enter the data you want
            to insert at the beginning\n");
        scanf("%d", &new);
        InsertAtBeginning(&head, new);
        break;
    case 2:
        printf("Enter the data and position
            at which you want to insert\n");
        scanf("%d%d", &new, &pos);
        InsertAtPosition(&head, new, pos);
        break;
    case 3:
        printf("Enter the data you want
            to insert at the end\n");
        scanf("%d", &new);
        InsertAtEnd(&head, new);
        break;
    case 4:
        printf("Created linked list is:\n");
        PrintList(head); break;
```

```
        case 5: return 0;
        default:
            printf("Invalid choice!\n");
    }
}
return 0;
}
```

Output:

```
Menu
1. Insert at beginning
2. Insert at a specific position
3. Insert at end
4. Display linked list
5. Exit
Enter your choice
1
Enter the data you want to insert at the beginning
11

Menu
1. Insert at beginning
2. Insert at a specific position
3. Insert at end
4. Display linked list
5. Exit
Enter your choice
3
Enter the data you want to insert at the end
13
```

Menu
1. Insert at beginning
2. Insert at a specific position
3. Insert at end
4. Display linked list
5. Exit
Enter your choice
2
Enter the data and position at which you want to insert
12 1

Menu
1. Insert at beginning
2. Insert at a specific position
3. Insert at end
4. Display linked list
5. Exit.
Enter your choice
4
Created linked list is:
11
12
13

Menu
1. Insert at beginning
2. Insert at a specific position
3. Insert at end
4. Display linked list
5. Exit
Enter your choice 5

DATE: / /

2) WAP to Implement Singly Linked List with following operations
a) Create a linked list.
b) Deletion of first element, specified element and last element in the list.
Display the contents of the linked list.

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

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

void InsertAtBeginning(Node **headref,
                       int newdata);
void DeleteAtBeginning(Node **headref);
void DeleteAtEnd(Node **headref);
void DeleteAtPosition(Node **headref, int pos);
void Printlist(Node *next);

void InsertAtBeginning(Node **headref,
                       int newdata){
    Node *newnode = (Node *) malloc(sizeof(Node));
    newnode->data = newdata;
    newnode->next = *headref;
    *headref = newnode;
}
```

```
void DeleteAtBeginning(Node **headref){
  Node *ptr;
  if (*headref == NULL)
    printf("\nList is empty");
  else{
  ptr = *headref;
  *headref = ptr->next;
  free(ptr);
  printf("\n Node deleted from the
          beginning ...");
  }
}

void DeleteAtEnd(Node **headref){
  Node *ptr, *ptr1;
  if (*headref == NULL)
    printf("\nlist is empty");
  else if((*headref)->next == NULL){
  free(*headref);
  *headref = NULL;
  printf("\n Only node of the list
                deleted...");
  else{
  ptr = *headref;
  while(ptr->next != NULL){
        ptr1 = ptr;
        ptr = ptr->next;
  }
  ptr1->next = NULL;
  free(ptr);
  printf("\n Deleted Node from the
                last ...");
}}
```

```
void DeleteAtPosition(Node **head_ref, int pos)
{
    Node *temp = *head_ref, *prev;
    if (temp == NULL) {
        printf("\nList is empty");
        return;
    }
    if (pos == 1) {
        *head_ref = temp -> next;
        free(temp);
        printf("\n Deleted node with
                position %d", pos);
        return; }
    for (int i=0; temp != NULL && i < pos-1; i++)
    {   prev = temp;
        temp = temp -> next;
    }
    if (temp == NULL){
        printf("\nPosition out of range");
        return; }
    prev -> next = temp -> next;
    free(temp);
    printf("\n Deleted node with position
                %d", pos);
}
void PrintList(Node *node){
    while (node != NULL) {
        printf("%d \n", node -> data);
        node = node -> next;
    }
}
```

```
int main(){
    int ch, new, pos;
    Node *head = NULL;
    while (ch != 6){
    printf("\nMenu\n");
    printf("1. Create a linked list \n");
    printf("2. Delete at beginning \n");
    printf("3. Delete at a specific position\n");
    printf("4. Delete at end \n");
    printf("5. Display linked list\n");
    printf("6. Exit\n");
    printf("Enter your choice\n");
    scanf("%d",&ch);
    switch (ch){
    case 1: printf("Enter the data you want
            to insert at the beginning \n");
            scanf("%d", &new);
            InsertAtBeginning (&head , new);
            break;
    case 2: DeleteAtBeginning(&head);
            break;
    case 3: printf("Enter the position at
            which you want to delete \n");
            scanf("%d", &pos);
            DeleteAtPosition (&head, pos);
            break;
    case 4: DeleteAtEnd(&head);
            break;
    case 5: printf("Created linked list is:\n");
            PrintList(head);
            break;
    case 6: return 0;
```

```
    default: printf("Invalid choice!\n");
    }
}return 0;
}
```

Output:

```
Menu
1. Create a linked list
2. Delete at beginning
3. Delete at a specific position
4. Delete at end
5. Display linked list
6. Exit.
Enter your choice
1
Enter the data you want to insert at the beginning
12

Menu
1. Create a linked list
2. Delete at beginning
3. Delete at a specific position
4. Delete at end
5. Display linked list
6. Exit
Enter your choice
2
Node deleted from the beginning...
```