4(a)_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 linked list.

```
#include <stdio.h>

#include <stdlib.h>


// Define node structure

struct Node {

   int data;

   struct Node* next;

};


// Function to create a new node

struct Node* createNode(int data) {

   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

   newNode->data = data;

   newNode->next = NULL;

   return newNode;

}


// Function to insert at the beginning

struct Node* insertAtBeginning(struct Node* head, int data) {

   struct Node* newNode = createNode(data);

   newNode->next = head;

   head = newNode;

   return head;

}


// Function to insert at the end
```

```
struct Node* insertAtEnd(struct Node* head, int data) {
    struct Node* newNode = createNode(data);
    if (head == NULL) {
        head = newNode;
        return head;
    }
    struct Node* temp = head;
    while (temp->next != NULL)
      temp = temp->next;
    temp->next = newNode;
    return head;
}

// Function to insert at a given position (1-based index)
struct Node* insertAtPosition(struct Node* head, int data, int pos) {
    if (pos == 1) {
        return insertAtBeginning(head, data);
    }

    struct Node* newNode = createNode(data);
    struct Node* temp = head;
    for (int i = 1; i < pos - 1 && temp != NULL; i++)
        temp = temp->next;
       if (temp == NULL) {
        printf("Position out of bounds.\n");
        free(newNode);
        return head;
    }
```

```
    newNode->next = temp->next;
    temp->next = newNode;
    return head;
}

// Function to display the linked list
void displayList(struct Node* head) {
    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }
    struct Node* temp = head;
    printf("Linked List: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main() {
    struct Node* head = NULL;
    int choice, data, pos;

    while (1) {
        printf("\n1. Insert at beginning\n2. Insert at end\n3. Insert at position\n4. Display\n5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
    switch (choice) {
        case 1:
            printf("Enter data: ");
            scanf("%d", &data);
            head = insertAtBeginning(head, data);
            break;
        case 2:
            printf("Enter data: ");
            scanf("%d", &data);
            head = insertAtEnd(head, data);
            break;
        case 3:
            printf("Enter position: ");
            scanf("%d", &pos);
            printf("Enter data: ");
            scanf("%d", &data);
            head = insertAtPosition(head, data, pos);
            break;
        case 4:
            displayList(head);
            break;
        case 5:
            exit(0);
        default:
            printf("Invalid choice.\n");
    }
}
```

```
return 0;
```

}

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Display
5. Exit
Enter your choice: 3
Enter position: 2
Enter data: 15

1. Insert at beginning
2. Insert at end
3. Insert at position
4. Display
5. Exit
Enter your choice: 4
Linked List: 5 -> 15 -> 10 -> 20 -> NULL

1. Insert at beginning
2. Insert at end
3. Insert at position
4. Display
5. Exit
Enter your choice: 5

Process returned 0 (0x0)   execution time : 64.294 s
Press any key to continue.
```

```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Display
5. Exit
Enter your choice: 2
Enter data: 10

1. Insert at beginning
2. Insert at end
3. Insert at position
4. Display
5. Exit
Enter your choice: 2
Enter data: 20

1. Insert at beginning
2. Insert at end
3. Insert at position
4. Display
5. Exit
Enter your choice: 1
Enter data: 5
```