4. WAP to Implement Singly Linked List with following operations a) Createalinkedlist. 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>
struct Node {
int data;
struct Node* next;
};
struct Node* createNode(int data) {
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
if (newNode == NULL) {
printf("Memory allocation failed\n");
return NULL;
}
newNode->data = data;
newNode->next = NULL;
return newNode;
}
struct Node* insertAtFirst(struct Node* head, int data) {
struct Node* newNode = createNode(data);
if (newNode == NULL) return head;
newNode->next = head;
return newNode;
}
struct Node* insertAtEnd(struct Node* head, int data) {
struct Node* newNode = createNode(data);
if (newNode == NULL) return head;
if (head == NULL) return newNode;
struct Node* temp = head;
while (temp->next != NULL) {
temp = temp->next;
}
temp->next = newNode;
return head;
}
struct Node* insertAtPosition(struct Node* head, int data, int position) {
if (position < 1) {
printf("Invalid position\n");
return head;
}
struct Node* newNode = createNode(data);
if (newNode == NULL) return head;
```

```
if (position == 1) {
newNode->next = head;
return newNode;
}
struct Node* temp = head;
int i;
for (i = 1; i < position - 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;
}
void displayList(struct Node* head) {
if (head == NULL) {
printf("List is empty\n");
return;
}
struct Node* temp = head;
while (temp != NULL) {
printf("%d ", temp->data);
temp = temp->next;
}
printf("\n");
}
void freeList(struct Node* head) {
struct Node* current = head;
while (current != NULL) {
struct Node* next = current->next;
free(current);
current = next;
}}
int main() {
struct Node* head = NULL;
int choice, data, position, numNodes;
printf("Enter number of initial nodes to create the list: ");
scanf("%d", &numNodes);
for (int i = 0; i < numNodes; i++) {
printf("Enter data for node %d: ", i + 1);
```

```
scanf("%d", &data);
head = insertAtEnd(head, data);
}
printf("List created.\n");
do {
printf("\nMenu:\n1. Insert at first position\n2. Insert at end\n3. Insert at position\n4. Display
list\n5.
Exit\n");
printf("Enter choice: ");
scanf("%d", &choice);
switch (choice) {
case 1:
printf("Enter data: ");
scanf("%d", &data);
head = insertAtFirst(head, data);
printf("Node inserted at first position.\n");
break;
case 2:
printf("Enter data: ");
scanf("%d", &data);
head = insertAtEnd(head, data);
printf("Node inserted at end.\n");
break;
case 3:
printf("Enter position (1-based): ");
scanf("%d", &position);
printf("Enter data: ");
scanf("%d", &data);
head = insertAtPosition(head, data, position);
printf("Node inserted at position %d.\n", position);
break;
case 4:
printf("List contents: ");
displayList(head);
break;
case 5:
printf("Exiting.\n");
break;
default:
printf("Invalid choice.\n");
}
} while (choice != 5);
freeList(head);
return 0; }
```

```
Enter number of initial nodes to create the list: 4
Enter data for node 1: 10
Enter data for node 2: 20
Enter data for node 3: 30
Enter data for node 4: 40
List created.

Menu:
1. Insert at first position
2. Insert at end
3. Insert at position
4. Display list
5. Exit
Enter choice: 1
Enter data: 25
Node inserted at first position.

Menu:
1. Insert at first position
2. Insert at end
3. Insert at position
4. Display list
5. Exit
Enter choice: 2
Enter data: 90
Node inserted at end.

Menu:
1. Insert at first position
2. Insert at end
3. Insert at position
4. Display list
5. Exit
Enter choice: 3
Enter position (1-based): 3
Enter data: 60
Node inserted at position 3.

Menu:
1. Insert at first position
2. Insert at end
3. Insert at position
4. Display list
5. Exit
Enter choice: 4
List contents: 25 10 60 20 30 40 90
```