⑤ WAP to create single linked list:-
→ Insertion at the beginning, middle and the end

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

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

struct ptr* createPtr(int data) {
    struct ptr* newPtr = (struct ptr*) malloc(sizeof(struct
    newPtr -> data = data;
    newPtr -> next = NULL;
    return newPtr;
}

struct ptr* insertAtBeginning(struct ptr* head, int data){
    struct ptr* newPtr = createPtr(data)
    newNode -> next = head;
    return newNode ptr;
}
void insertAtPosition(struct ptr* head, int data, int position) {
    struct Node* newNodeptr = createPtr(data);
    struct ptr* current = head;

    for (int i=0; i < position - 1; i++) {
        if (current == NULL) {
            printf("Position out of bounds\n");
            return;
        }
        current = current -> next;
    }
```

```
    newPtr->next = current->next;
    current->next = newPtr;
}

struct Ptr* insertAtEnd(struct Ptr* head, int data){
    struct Ptr* newPtr = createPtr(data);
    if(head==NULL){
        return newPtr;
}

struct Ptr* current = head;
while(current->next != NULL){
        current = current->next;
}
current->next = newPtr;
return head;

void display(struct Ptr* head){
    struct Ptr* current = head;
    while(current != NULL){
        printf("%d -> ", current->data);
        current = current->next;
}
printf("NULL\n");
}
int main(){
    struct Ptr* linkedList = NULL;

    linkedList = insertAtEnd(linkedList, 1);
    linkedList = insertAtEnd(linkedList, 2);
    linkedList = insertAtEnd(linkedList, 3);

    display(linkedList);
    linkedList = insertAtBeginning(linkedList, 0);
    display(linkedList);
```

```
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
6.-1 to end
1
enter the number of elements:
2
Enter the element 1:
3
Enter the element 2:
4
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
6.-1 to end
2
3
4
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
6.-1 to end
3
Enter the new element:
5
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
6.-1 to end
2
5
3
4
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
```

```
4.insert at end
5.insert at position
6.-1 to end
4
Enter the new element:
6
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
6.-1 to end
2
5
3
4
6
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
6.-1 to end
5
enter the position:
2
Enter the new element:
9
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
6.-1 to end
2
5
9
3
4
6
Enter operation:
1.create
2.display
3.insert at beginnning
4.insert at end
5.insert at position
6.-1 to end
```

insertAtPosition(linkedList, 1, 3);
display(linkedList);

return 0;
}

6. WAP to create single linked list and show solution at the beginning, in the middle and at the end.

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

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

struct node *head = NULL, *newnode, *temp;

void create()
{
    int i, n;
    printf("Enter the no of elements: \n");
    scanf("%d", &n);

    for(i=0; i<n; i++)
    {
        newnode = (struct node *) malloc(sizeof(struct node));
        printf("Enter the %d element: \n", i+1);
        scanf("%d", &newnode->data);
        newnode->next = NULL;

        if(head == NULL)
        {
            temp = head = newnode;
        }
```

```
else {
        temp -> next = newnode;
        temp = newnode;
      }
    }
}

void display ()
 {
   temp = head;
   printf (" the elements are: \n");
   while (temp != NULL)
     {
       printf (" %d\n", temp -> data);
       temp = temp -> next;
     }
 }

void delete_beg()
{
  temp = head;
  if (head == NULL)
   {
    printf (" list is empty ");
   }
   else
   {
    head = temp -> next;
      free (temp);
   }
}
```

```
void delete_end(){
    struct node *prenode;
    temp = head;
    while (temp -> next != NULL)
    {
        prenode = temp;
        temp = temp -> next;
    }
    if (temp = head)
    {
        head = NULL;
    }
    else
    {
        prenode -> next = NULL;
    free(temp);
}
void delete_pos()
{
    struct node * next node;
    int pos, i = 1;
    temp = head;
    printf("Enter position \n");
    scanf("%d", &pos);
    while (i < pos)
    {
        temp = temp -> next;
        i++;
    }
    next node = temp -> next;
    temp -> next = next node -> next;
    free(next node);
}
void main()
```

```
{
  int choice;
  while (1)
  {
    printf("Enter operation : \n 1. create \n 2. display \n 3.
        delete at beginning \n 4. delete at end \n 5. delete at
        a position \n 6. -1 to end \n");
    scanf("%d", &choice);
    if (choice == -1)
    {
      printf("operation completed ! \n");
      break;
    }
    else switch choice();
        case 1: create();
        break;
        case 2: display();
        break;
        case 3: delete_beg();
        break;
        case 4: delete_end();
        break;
        case 5: delete_pos();
        break;
        default: printf("Invalid output \n");
      }
    }
  }
}
```

```
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
4
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
2
The elements are:
3
4
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
5
enter the position:
2
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
2
The elements are:
3
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
```

```
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
1
enter the number of elements:
4
Enter the element 1:
2
Enter the element 2:
3
Enter the element 3:
4
Enter the element 4:
5
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
2
The elements are:
2
3
4
5
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
3
Enter operation:
1.create
2.display
3.delete at beginnning
4.delete at end
5.delete at position
6.-1 to end
2
The elements are:
3
4
5
```