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

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

struct node * create (int a)
{
    struct node* newnode = (struct node*) malloc (sizeof(struct node));
    newnode->data=a;
    newnode->next=0;
    return newnode;
};
struct node * insertatbeg(int a, struct node* head)
{
    struct node * newnode =create(a);
    newnode->next=head;
    head=newnode;
    return head;
};
struct node * insertatpos(int a, struct node *head, int pos )
{
    struct node*temp=head;
    int i=1;
    while(i>pos-1)
    {
        temp=temp->next;
    }
    struct node* newnode = create(a);
    newnode->next=temp->next;
    temp->next=newnode;

};
struct node * insertatend(int a, struct node* head)
{
    struct node*temp=head;
    while(temp->next!=NULL)
    {
        temp=temp->next;
    }
    struct node* newnode = create(a);
    temp->next=newnode;
```

```
struct node * insertatend(int a, struct node* head)
{
    struct node*temp=head;
    while(temp->next!=NULL)
    {
    struct node* newnode = create(a);
};
void display( struct node*head)
{
    struct node* temp= head;
    while(temp->next!=NULL)
    {
    printf("%d \t -> \t",temp->data);
    temp=temp->next;
    }
    printf("%d",temp->data);
}
void main()
{
    int choice, pos,value;
    struct node*head=NULL;
    printf("enter-1.beg-2.end,-3.pos- 4.display 5.exit");
    do{
        printf("\n Enter choice \t");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("\n Enter the element to be inserted at beginning \t");
                   scanf("%d",&value);
                   head=insertatbeg(value,head);
                   break;
            case 2: printf("\n Enter the element to be inserted at end \t");
                   scanf("%d",&value);
                   insertatend(value,head);
                   break;
            case 3:printf("\n Enter the element to be inserted at position \t");
                   scanf("%d",&value);
```

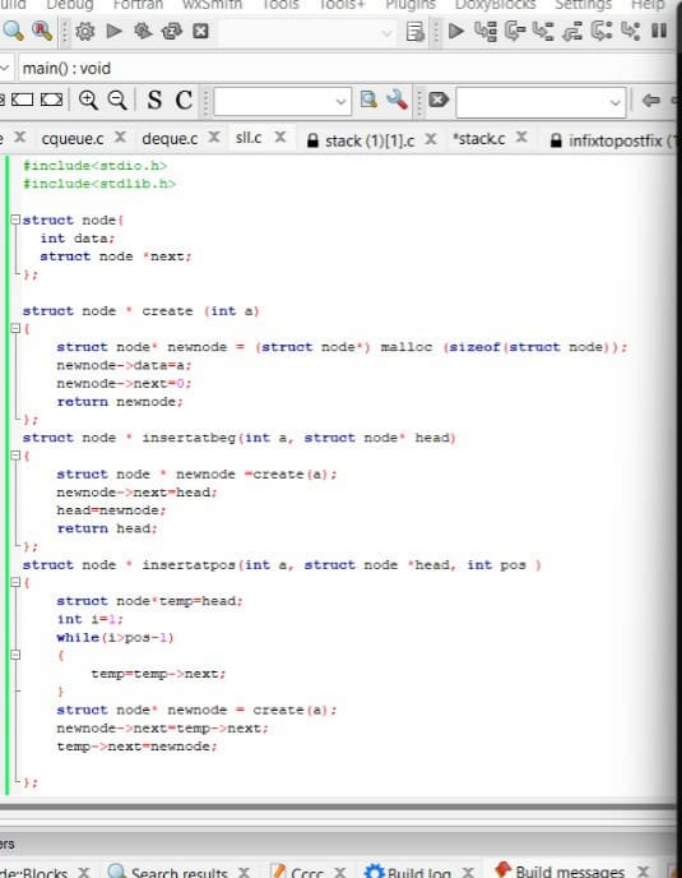
```
{
    struct node* temp= head;
    while(temp->next!=NULL)
    {
        printf("%d \t -> \t",temp->data);
        temp=temp->next;
    }
    printf("%d",temp->data);
}

void main()
{
    int choice, pos,value;
    struct node*head=NULL;
    printf("enter 1.beg 2.end, 3.pos  4.display 5.exit");
    do{

        printf("\n Enter choice \t");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:printf("\n Enter the element to be inserted at beginning \t");
                    scanf("%d",&value);
                    head=insertatbeg(value,head);
                    break;
            case 2: printf("\n Enter the element to be inserted at end \t");
                    scanf("%d",&value);
                    insertatend(value,head);
                    break;
            case 3:printf("\n Enter the element to be inserted at position \t");
                    scanf("%d",&value);
                   printf("\n Enter pos");
                    scanf("%d",&pos);
                    insertatpos(value,head,pos);
                    break;
            case 4: display(head);
                    break;
            case 5: exit(0);

        }
    }while(choice != 6);

}
```

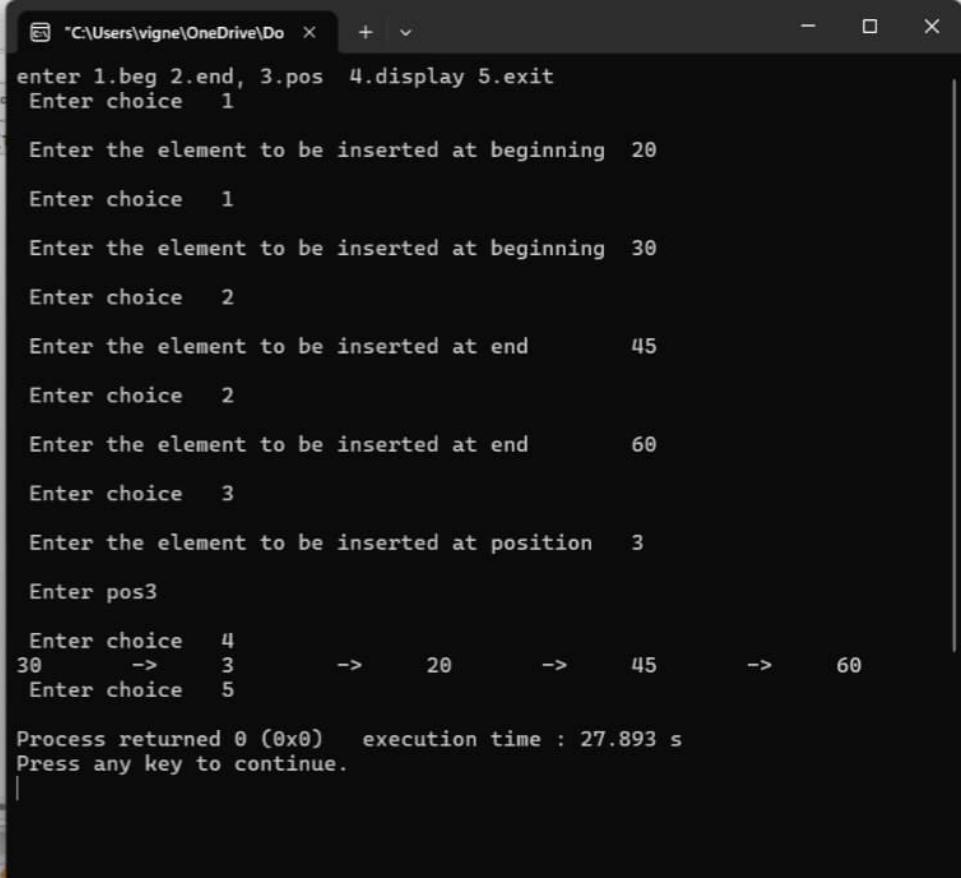


```
#include<stdio.h>
#include<stdlib.h>

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

struct node * create (int a)
{
    struct node* newnode = (struct node*) malloc (sizeof(struct node));
    newnode->data=a;
    newnode->next=0;
    return newnode;
};
struct node * insertatbeg(int a, struct node* head)
{
    struct node * newnode =create(a);
    newnode->next=head;
    head=newnode;
    return head;
};
struct node * insertatpos(int a, struct node *head, int pos )
{
    struct node*temp=head;
    int i=1;
    while(i>pos-1)
    {
        temp=temp->next;
    }
    struct node* newnode = create(a);
    newnode->next=temp->next;
    temp->next=newnode;

};
```



```
"C:\Users\vigne\OneDrive\Do

enter 1.beg 2.end, 3.pos  4.display 5.exit
 Enter choice   1

 Enter the element to be inserted at beginning  20

 Enter choice   1

 Enter the element to be inserted at beginning  30

 Enter choice   2

 Enter the element to be inserted at end        45

 Enter choice   2

 Enter the element to be inserted at end        60

 Enter choice   3

 Enter the element to be inserted at position   3

 Enter pos3

 Enter choice   4
30       ->     3        ->     20       ->     45       ->     60
 Enter choice   5

Process returned 0 (0x0)   execution time : 27.893 s
Press any key to continue.
```

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

```c
#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");
        exit(1);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

struct node *insertAtBeginning(struct node *head, int data) {
    struct node *newNode = createNode(data);
    if (head == NULL) {
        head = newNode;
    } else {
        newNode->next = head;
        head = newNode;
    }
    return head;
}
```

```c
struct node *deleteAtBeginning(struct node *head) {
    if (head == NULL) {
        printf("List is empty\n");
    } else {
        struct node *temp = head;
        head = head->next;
        free(temp);
        printf("Node deleted from the beginning\n");
    }
    return head;
}


struct node *deleteAtEnd(struct node *head) {
    if (head == NULL) {
        printf("List is empty\n");
    } else if (head->next == NULL) {
        free(head);
        head = NULL;
        printf("Node deleted from the end\n");
    } else {
        struct node *temp = head;
        struct node *prev = NULL;
        while (temp->next != NULL) {
            prev = temp;
            temp = temp->next;
        }
        prev->next = NULL;
        free(temp);
        printf("Node deleted from the end\n");
    }
    return head;
}
```

```c
struct node *deleteAtPosition(struct node *head, int position) {
    if (head == NULL) {
        printf("List is empty\n");
    } else if (position == 1) {
        head = deleteAtBeginning(head);
    } else {
        struct node *temp = head;
        struct node *prev = NULL;
        int count = 1;
        while (temp != NULL && count < position) {
            prev = temp;
            temp = temp->next;
            count++;
        }
        if (temp == NULL) {
            printf("Position not found\n");
        } else {
            prev->next = temp->next;
            free(temp);
            printf("Node deleted from position %d\n", position);
        }
    }
    return head;
}

void display(struct node *head) {
    printf("Linked list: ");
    struct node *temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    struct node *head = NULL;
    head = insertAtEnd(head, 10);
    head = insertAtEnd(head, 20);
    head = insertAtEnd(head, 30);
    head = insertAtEnd(head, 40);
```

```c
int main() {
    struct node *head = NULL;
    head = insertAtEnd(head, 10);
    head = insertAtEnd(head, 20);
    head = insertAtEnd(head, 30);
    head = insertAtEnd(head, 40);
    head = insertAtEnd(head, 50);

    int choice, position;

    do {
        printf("\n1. Delete at beginning\n");
        printf("2. Delete at end\n");
        printf("3. Delete at a specific position\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                head = deleteAtBeginning(head);
                break;
            case 2:
                head = deleteAtEnd(head);
                break;
            case 3:
                printf("Enter the position to delete: ");
                scanf("%d", &position);
                head = deleteAtPosition(head, position);
                break;
            case 4:
                display(head);
                break;
            case 5:
                printf("Exiting the program\n");
                break;
        }
    } while(choice != 5 );
}
```

```
"C:\Users\vigne\OneDrive\Do

1. Delete at beginning
2. Delete at end
3. Delete at a specific position
4. Display
5. Exit
Enter your choice: 4
Linked list: 10 20 30 40 50

1. Delete at beginning
2. Delete at end
3. Delete at a specific position
4. Display
5. Exit
Enter your choice: 1
Node deleted from the beginning

1. Delete at beginning
2. Delete at end
3. Delete at a specific position
4. Display
5. Exit
Enter your choice: 2
Node deleted from the end

1. Delete at beginning
2. Delete at end
3. Delete at a specific position
4. Display
5. Exit
Enter your choice:
4
Linked list: 20 30 40

1. Delete at beginning
2. Delete at end
3. Delete at a specific position
4. Display
5. Exit
Enter your choice: 3
Enter the position to delete: 2
Node deleted from position 2

1. Delete at beginning
2. Delete at end
3. Delete at a specific position
4. Display
5. Exit
Enter your choice: 4
Linked list: 20 40

1. Delete at beginning
2. Delete at end
3. Delete at a specific position
4. Display
5. Exit
Enter your choice:
```

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