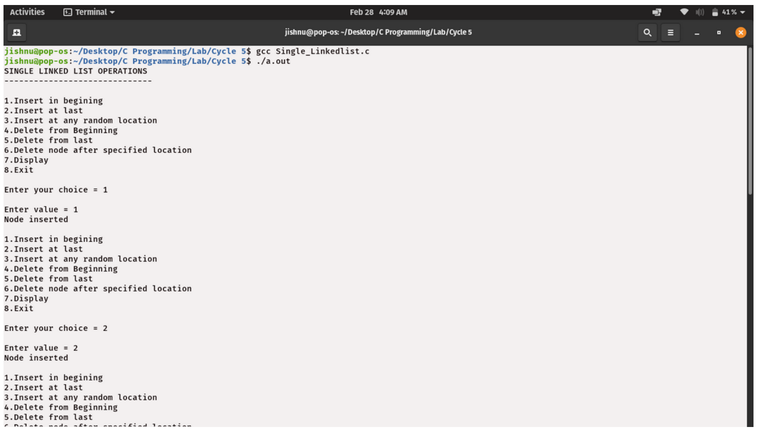
**SINGLE LINKED LIST**

```
#include<stdio.h>
#include<stdlib.h>
struct node{
	int data;
	struct node *next;
};
struct node *head;
void insert_Front();
void insert_End();
void insert_Any();
void delete_Front();
void delete_End();
void delete_Any();
void print();
void main(){
	printf("SINGLE LINKED LIST OPERATIONS\n");
		for(int k=0;k<29;k++){
			printf("%c",'-');
		}printf("\n");
	int choice = 1;
	while(choice<9){
		printf("1.insert at beginning\n2.insert at end\n3.insert at any position\n4.delete from
beginning\n5.delete from end\n6.delete from any position\n7.print\n8.exit\n\n");
		printf("Enter your choice = ");
		scanf("%d",&choice);
	switch(choice){
		case 1: insert_Front();
		break;
		case 2: insert_End();
		break;
		case 3: insert_Any();
		break;
		case 4: delete_Front();
		break;
		case 5: delete_End();
		break;
		case 6: delete_Any();
		break;
		case 7: print();
		break;
		case 8: exit(0);
		break;
		default:printf("invalid key...program terminated !!!\n");
		exit(0);
	}
}}
void insert_Front(){
	struct node *ptr;
	int item;
	ptr = (struct node *) malloc(sizeof(struct node *));
	if(ptr == NULL){
		printf("overflow\n");
	}else{
		printf("\nEnter value = ");
		scanf("%d",&item);
		ptr->data = item;
```

```
                ptr->next = head;
                head = ptr;
                printf("Node inserted successfully\n\n");
        }
}
void insert_End(){
        struct node *ptr,*temp;
        int item;
        ptr = (struct node*)malloc(sizeof(struct node));
        if(ptr == NULL){
                printf("overflow\n");
        }else{
                printf("Enter value = ");
                scanf("%d",&item);
                ptr->data = item;
                if(head == NULL){
                        ptr -> next = NULL;
                        head = ptr;
                        printf("Node inserted successfully\n\n");
                }else{
                        temp = head;
                        while (temp -> next != NULL){
                                temp = temp -> next;
                        }
                        temp->next = ptr;
                        ptr->next = NULL;
                        printf("Node inserted successfully\n\n");
                }
        }
}
void insert_Any(){
        int i,l,item;
        struct node *ptr, *temp;
        ptr = (struct node *) malloc (sizeof(struct node));
        if(ptr == NULL){
                printf("overflow\n");
        }else{
                printf("\nEnter element value = ");
                scanf("%d",&item);
                ptr->data = item;
                printf("Enter the location after which you want to insert = ");
                scanf("%d",&l);
                temp=head;
                for(i=0;i<l;i++){
                        temp = temp->next;
                        if(temp == NULL){
                                printf("insertion failed\n\n");
                                return;
                        }
                }
                ptr ->next = temp ->next;
                temp ->next = ptr;
                printf("Node inserted successfully\n\n");
        }
}
void delete_Front(){
        struct node *ptr;
```

```
	if(head == NULL){
		printf("underflow\n\n");
	}else{
		ptr = head;
		head = ptr->next;
		free(ptr);
		printf("Node deleted from the begining\n\n");
	}
}
void delete_End(){
	struct node *ptr,*ptr1;
	if(head == NULL){
		printf("underflow\n\n");
	}
	else if(head -> next == NULL){
		head = NULL;
		free(head);
		printf("Only node of the list deleted\n\n");
	}else{
		ptr = head;
		while(ptr->next != NULL){
			ptr1 = ptr;
			ptr = ptr ->next;
		}
		ptr1->next = NULL;
		free(ptr);
		printf("Deleted Node from the last ...\n\n ");
	}
}
void delete_Any(){
	struct node *ptr,*ptr1;
	int l,i;
	printf("Enter the location to delete node = ");
	scanf("%d",&l);
	ptr=head;
	for(i=0;i<l;i++){
		ptr1 = ptr;
		ptr = ptr->next;
		if(ptr == NULL){
			printf("deletion failed\n\n");
			return;
		}
	}
	ptr1 ->next = ptr ->next;
	free(ptr);
	printf("Deleted node %d ",l+1);
}
void print(){
	struct node *ptr;
	ptr = head;
	if(ptr == NULL){
		printf("underflow\n");
	}else{
		printf("The current Linked List\n\n");
		while (ptr!=NULL){
			printf("%d\n",ptr->data);
```

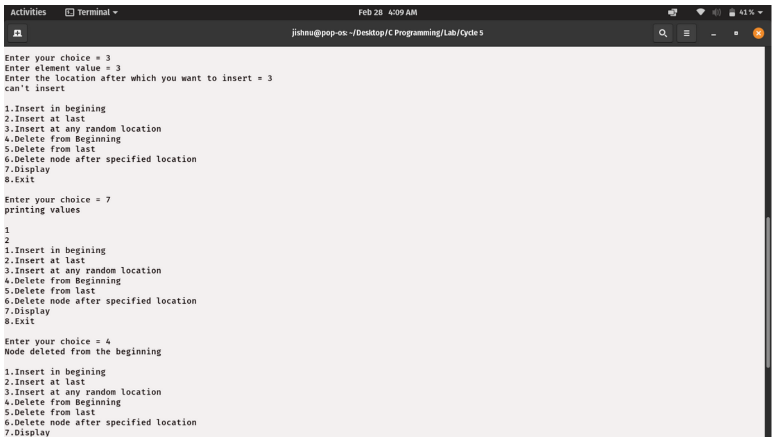
```
                ptr = ptr -> next;
            }
            printf("\n");
        }
}
```



Activities   Terminal ▾   Feb 28  4:09 AM   41 %

jishnu@pop-os: ~/Desktop/C Programming/Lab/Cycle 5

```
jishnu@pop-os:~/Desktop/C Programming/Lab/Cycle 5$ gcc Single_Linkedlist.c
jishnu@pop-os:~/Desktop/C Programming/Lab/Cycle 5$ ./a.out
SINGLE LINKED LIST OPERATIONS
------------------------------

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Display
8.Exit

Enter your choice = 1

Enter value = 1
Node inserted

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Display
8.Exit

Enter your choice = 2

Enter value = 2
Node inserted

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
```



Activities   Terminal ▾   Feb 28  4:09 AM   41 %

jishnu@pop-os: ~/Desktop/C Programming/Lab/Cycle 5

```
Enter your choice = 3
Enter element value = 3
Enter the location after which you want to insert = 3
can't insert

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Display
8.Exit

Enter your choice = 7
printing values

1
2
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Display
8.Exit

Enter your choice = 4
Node deleted from the beginning

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete node after specified location
7.Display
```