WEEK 1

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

/* structure of node */
struct node {
	int data;
	struct node *next;
};
typedef struct node *NODE;

/* CREATING THE NODE */
NODE createNode(){
	NODE ptr;
	ptr=(NODE)malloc(sizeof(struct node));
	ptr->next=NULL;
	return ptr;
}


/* INSERTING AT THE END OF THE LIST */
NODE insertAtEnd(NODE head, int value){
	NODE newNode=createNode();
	NODE dhead;
	newNode->data=value;
	if(head==NULL)
		head=newNode;
	else{
		dhead=head;
		while(dhead->next!=NULL)
			dhead=dhead->next;
		dhead->next=newNode;
	}
```

```
        return head;
}
```

/* INSERTING AT THE SPECIFIC POSITION */

```
NODE insertAtPosition(NODE head, int value, int position){
        int count=0;
        NODE dhead, previous=NULL, newNode=createNode();
        newNode->data=value;
        if(head==NULL){
                printf("\n list is empty.");
        }
        else
        {
                dhead=head;
                count=1;

                if(position==1)
                        head=insertAtEnd(head, value);
                else{
                        while(dhead!=NULL && count < position){
                                previous=dhead;
                                dhead=dhead->next;
                                count++;
                        }
                        if(dhead==NULL){
                                dhead->next = newNode;
                        }
                        else{
                                previous->next=newNode;
                                newNode->next=dhead;
                        }
                }
        }
        return head;
}
```

/* DELETE FROM THE FRONT OF THE LIST */

```
NODE deleteAtFirst(NODE head){
        NODE dhead;
        if(head==NULL)
                printf("\n List is empty\n");
        else{
                if(head->next==NULL){
                        free(head);
                        head=NULL;
                }
                else
                {
                        dhead=head;
                        head=head->next;
                        free(dhead);
                        dhead=NULL;
                }
        }
        return head;
}
```

/* DISPLAYING THE LIST */

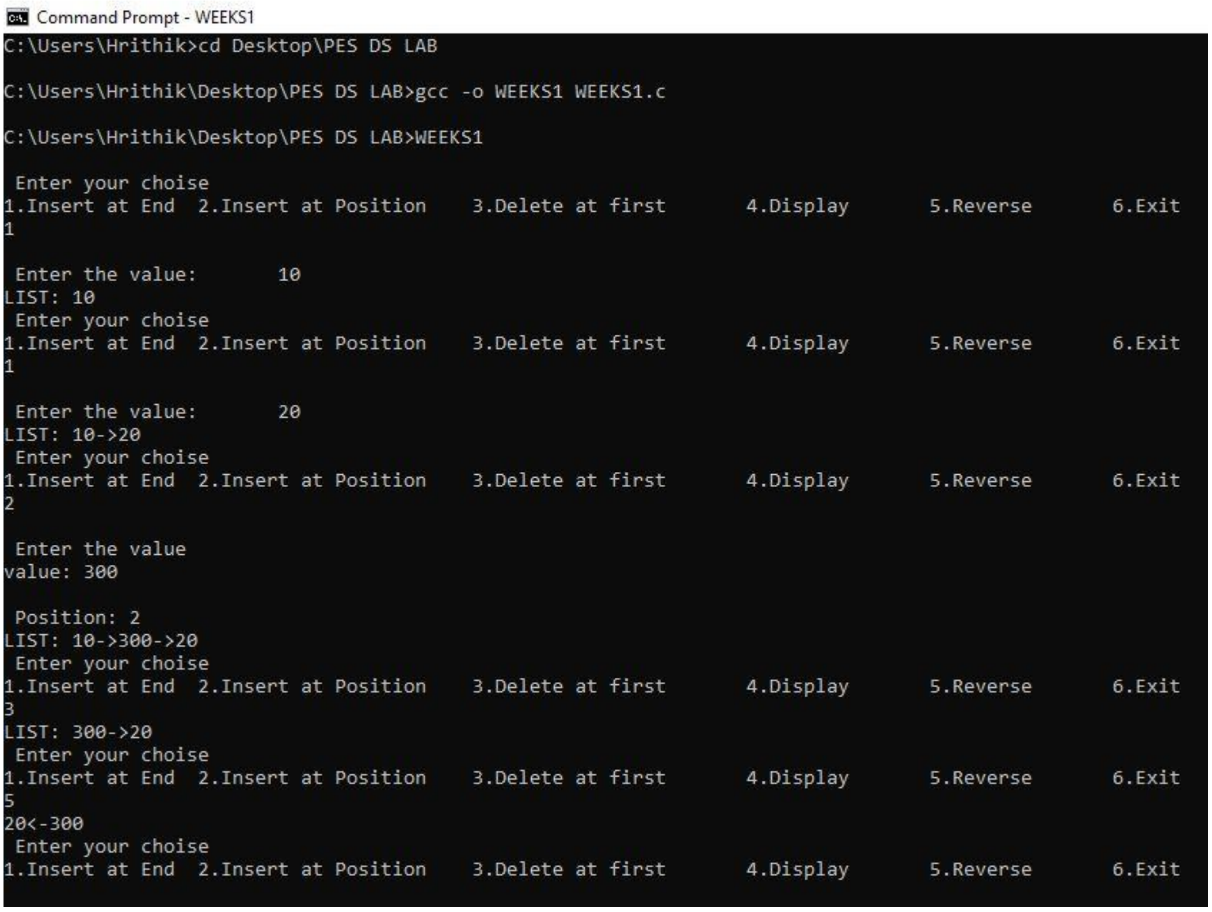
```
void display(NODE head){
        NODE dhead;
        if(head==NULL){
                printf("\nList is empty");
        }
        else{
                printf("LIST: ");
                dhead=head;
                while(dhead!=NULL)
                {
                        printf("%d",dhead->data);
                        if(dhead->next!=NULL)
                                printf("->");
                        dhead=dhead->next;
                }
        }
}
```

/* REVERSING THE LIST */

```
void reverseList(NODE head){
        NODE current, temp=NULL;
        if(head==NULL)
        {
                printf("\n List is empty\n");
        }
        else{
                while(temp!=head)
                {
                        current=head;
                        while(current->next != NULL && current->next != temp)
                                current= current->next;
                        if(current->next!=NULL){
                                printf("<-");
                        }
                        printf("%d",current->data);
                        temp=current;
                }
        }
}

int main()
{
        NODE head=NULL;
        int value, position, choise;
        while(choise!=6){
                printf("\n Enter your choise\n");
                printf("1.Insert at End\t 2.Insert at Position\t 3.Delete at first\t 4.Display\t 5.Reverse\t
6.Exit\n");
                scanf("%d",&choise);
                switch(choise)
                {
                        case 1:
                        printf("\n Enter the value:\t");
                        scanf("%d",&value);
                        head=insertAtEnd(head, value);
                        display(head);
                        break;

                        case 2:
                        printf("\n Enter the value\n");
                        printf("value: ");
                        scanf("%d",&value);
                        printf("\n Position: ");
                        scanf("%d", &position);
                        head=insertAtPosition(head, value, position);
                        display(head);
                        break;

                        case 3:
                        head=deleteAtFirst(head);
                        display(head);
                        break;

                        case 4:
                        display(head);
                        break;

                        case 5:
```

```
                        reverseList(head);
                        break;
                }
        }
}
```

OUTPUT



Command Prompt - WEEKS1

```
C:\Users\Hrithik>cd Desktop\PES DS LAB

C:\Users\Hrithik\Desktop\PES DS LAB>gcc -o WEEKS1 WEEKS1.c

C:\Users\Hrithik\Desktop\PES DS LAB>WEEKS1

 Enter your choise
1.Insert at End  2.Insert at Position    3.Delete at first       4.Display       5.Reverse       6.Exit
1

 Enter the value:       10
LIST: 10
 Enter your choise
1.Insert at End  2.Insert at Position    3.Delete at first       4.Display       5.Reverse       6.Exit
1

 Enter the value:       20
LIST: 10->20
 Enter your choise
1.Insert at End  2.Insert at Position    3.Delete at first       4.Display       5.Reverse       6.Exit
2

 Enter the value
value: 300

 Position: 2
LIST: 10->300->20
 Enter your choise
1.Insert at End  2.Insert at Position    3.Delete at first       4.Display       5.Reverse       6.Exit
3
LIST: 300->20
 Enter your choise
1.Insert at End  2.Insert at Position    3.Delete at first       4.Display       5.Reverse       6.Exit
5
20<-300
 Enter your choise
1.Insert at End  2.Insert at Position    3.Delete at first       4.Display       5.Reverse       6.Exit
```