/*Ques-PROGRAM TO ENTER THE KEY VALUE BY THE USER AND IF THE KEY VALUE IS FOUND UPDATE THE LIST BY DELETING THE KEY VALUE(BY SINGLY LL).*/

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

typedef struct node {

	int data ;
	struct node *next ;
}	Node ;

void display( Node* head ){

	while( head != NULL ){

		printf( "%d ", head-> data ) ;
		head = head-> next ;
	}
	printf( "\n" ) ;
}

Node* create( int val ){

	Node* temp = (Node*) malloc( sizeof( Node ) ) ;
	temp-> next = NULL ;
	temp-> data = val ;

	return temp ;
}
```

```
Node* add_End ( Node* head, int val ){

	//	create a new node
	Node* temp = create( val ) ;

	//	If List is Empty
	if( head == NULL ){
		return temp ;
	}

	//	Saving pointer to 1st node
	Node* ret = head ;

	//	point head at last position to add node
	while( head-> next != NULL ){
		head = head-> next ;
	}
	head-> next = temp ;

	return ret ;
}

Node* add_Beg( Node* head, int val ){

	//	create a new node
	Node* temp = create( val ) ;

	//	Point new node's next to current head
	temp-> next = head ;
```

```
        head = temp ;

        //        return new head
        return head ;
}

Node* del_key( Node* head, int key ){
    if( head-> data == key ){
        Node* tmp = head ;
        head = head-> next ;

        free( tmp ) ;
        return head ;
    }

    Node* ret = head ;
    while( head-> next != NULL && head-> next-> data != key ){
        head = head-> next ;
    }
    //  No data is found == key
    if( head-> next == NULL ){
        return ret ;
    }

    //  Now next node is the node to delete
    Node* tmp = head-> next ;
    head-> next = tmp-> next ;

    free( tmp ) ;
    return ret ;
```

```
}

int main(){

	Node *head = NULL ;

	head = add_End( head, 23 ) ;
	head = add_End( head, 43 ) ;
	head = add_End( head, 53 ) ;
	head = add_Beg( head, 13 ) ;

	display( head ) ;

	int key ;
	scanf( "%d", &key ) ;

	head = del_key( head, key ) ;

	display( head ) ;

	return 0 ;
}
```

/*Ques-PROGRAM TO ENTER THE KEY VALUE BY THE USER AND IF THE KEY VALUE IS FOUND UPDATE THE LIST BY DELETING THE KEY VALUE(BY SINGLY CIRCULAR LL).*/

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

typedef struct node {

	int data ;
	struct node *next ;
}	Node ;

void display( Node* head ){

   Node* ret = head ;
   printf( "%d ", head-> data ) ;
   head = head-> next ;
	while( head != ret ){

		printf( "%d ", head-> data ) ;
		head = head-> next ;
	}
	printf( "\n" ) ;
}

Node* create( int val ){

	Node* temp = (Node*) malloc( sizeof( Node ) ) ;
	temp-> next = temp ;
	temp-> data = val ;

	return temp ;
}

Node* add_End ( Node* head, int val ){

	//	create a new node
	Node* temp = create( val ) ;

	//	If List is Empty
	if( head == NULL ){
		return temp ;
	}
```

```
        //      Saving pointer to 1st node
        Node* ret = head ;

        //      point head at last position to add node
        while( head-> next != ret ){
                head = head-> next ;
        }
        head-> next = temp ;
        temp-> next = ret ;

        return ret ;
}

Node* add_Beg( Node* head, int val ){

        //      create a new node
        Node* temp = create( val ) ;

   Node* ret = head ;
   //   point head at last position to add node
        while( head-> next != ret ){
                head = head-> next ;
        }

   head-> next = temp ;
   temp-> next = ret ;

        //      return new head
        return temp ;
}

Node* del_key( Node* head, int key ){
   if( head-> data == key ){

      Node* ret = head ;
      while( head-> next != ret ){
         head = head-> next ;
      }
      //  Point last node to ret-> next
      head-> next = ret-> next ;
      head = head-> next ;

      free( ret ) ;
      return head ;
   }

   Node* ret = head ;
   while( head-> next != ret && head-> next-> data != key ){
```

```
        head = head-> next ;
    }
    //  No data is found == key
    if( head-> next == ret ){
        return ret ;
    }

    //  Now next node is the node to delete
    Node* tmp = head-> next ;
    head-> next = tmp-> next ;

    free( tmp ) ;
    return ret ;
}

int main(){

        Node *head = NULL ;

        head = add_End( head, 23 ) ;
        head = add_End( head, 43 ) ;
        head = add_End( head, 53 ) ;
        head = add_Beg( head, 13 ) ;

        display( head ) ;

        int key ;
        scanf( "%d", &key ) ;

        head = del_key( head, key ) ;

        display( head ) ;

        return 0 ;
}
```

/*Ques-PROGRAM TO ENTER THE KEY VALUE BY THE USER AND IF THE KEY VALUE IS FOUND UPDATE THE LIST BY DELETING THE KEY VALUE(BY DOUBLY
LL).*/

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

typedef struct node {

	int data ;
	struct node *next ;
	struct node *prev ;
}	Node ;

void display( Node* head ){

	while( head != NULL ){

		printf( "%d ", head-> data ) ;
		head = head-> next ;
	}
	printf( "\n" ) ;
}

Node* create( int val ){

	Node* temp = (Node*) malloc( sizeof( Node ) ) ;
	temp-> next = NULL ;
	temp-> prev = NULL ;
	temp-> data = val ;

	return temp ;
}

Node* add_End ( Node* head, int val ){

	//	create a new node
	Node* temp = create( val ) ;
```

```
        //      If List is Empty
        if( head == NULL ){
                return temp ;
        }

        //      Saving pointer to 1st node
        Node* ret = head ;

        //      point head at last position to add node
        while( head-> next != NULL ){
                head = head-> next ;
        }
        head-> next = temp ;
        temp-> prev = head ;

        return ret ;
}

Node* del_key( Node* head, int key ){
   if( head-> data == key ){

      Node* ret = head ;
      head = head-> next ;

      free( ret ) ;
      return head ;
   }

   Node* ret = head ;
   while( head-> next != NULL && head-> next-> data != key ){
      head = head-> next ;
   }
   //  No data is found == key
   if( head-> next == NULL ){
      return ret ;
   }

   //  Now next node is the node to delete
   Node* tmp = head-> next ;
   head-> next = tmp-> next ;
   head-> next-> prev = head ;
```

```
    free( tmp ) ;

    return ret ;
}

int main(){

        Node *head = NULL ;

        head = add_End( head, 23 ) ;
        head = add_End( head, 43 ) ;
        head = add_End( head, 53 ) ;
        head = add_End( head, 13 ) ;

        display( head ) ;

        int key ;
        scanf( "%d", &key ) ;

        head = del_key( head, key ) ;

        display( head ) ;

        return 0 ;
}
```