

```

#include <stdio.h>

#include <stdlib.h> // for malloc() and free()


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


// Function to create linked list
node* create() {
    node *head = NULL, *p, *q;
    int i, n;

    printf("Enter the number of nodes:\n");
    scanf("%d", &n);

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

        if (head == NULL) {
            head = q = p;
        } else {
            q->next = p;
            q = p;
        }
    }
    return head;
}

```

// Insert at front

```
node* insert_front(node *head, int x) {  
    node *p = (node*) malloc(sizeof(node));  
    p->data = x;  
    p->next = head;  
    return p;  
}
```

// Insert at rear

```
node* insert_rear(node *head, int x) {  
    node *p = (node*) malloc(sizeof(node));  
    p->data = x;  
    p->next = NULL;
```

```
    if (head == NULL) {  
        return p;  
    }
```

```
    node *q = head;  
    while (q->next != NULL) {  
        q = q->next;  
    }  
    q->next = p;  
    return head;  
}
```

// Delete front node

```
node* delete_front(node *head) {  
    if (head == NULL) {  
        printf("List is empty\n");
```

```
    return head;
}
```

```
node *p = head;
head = head->next;
printf("\nItem deleted: %d\n", p->data);
free(p);
return head;
}
```

// Delete rear node

```
node* delete_rear(node *head) {
    if (head == NULL) {
        printf("List is empty\n");
        return head;
    }
```

```
    if (head->next == NULL) {
        printf("\nItem deleted: %d\n", head->data);
        free(head);
        return NULL;
    }
```

```
    node *q = head;
    while (q->next->next != NULL) {
        q = q->next;
    }
```

```
    node *p = q->next;
    printf("\nItem deleted: %d\n", p->data);
    free(p);
```

```
q->next = NULL;
return head;
}
```

// Delete a node after a given value

```
node* delete_after(node *head, int x) {
    if (head == NULL) {
        printf("List is empty\n");
        return head;
    }

    node *p = head;
    while (p != NULL && p->data != x) {
        p = p->next;
    }

    if (p == NULL || p->next == NULL) {
        printf("No node found after %d\n", x);
        return head;
    }

    node *q = p->next;
    p->next = q->next;
    printf("Item deleted: %d\n", q->data);
    free(q);

    return head;
}
```

// Display linked list

```
void display(node *head) {
```

```
if (head == NULL) {  
    printf("List is empty\n");  
    return;  
}
```

```
printf("\nElements in the list:\n");  
for (node *p = head; p != NULL; p = p->next) {  
    printf("%d\t", p->data);  
}  
printf("\n");  
}
```

```
int main() {  
    node *head = NULL;  
  
    head = create();  
    display(head);  
  
    head = insert_front(head, 10);  
    display(head);  
  
    head = insert_rear(head, 20);  
    display(head);  
  
    head = delete_front(head);  
    display(head);  
  
    head = delete_rear(head);  
    display(head);  
  
    head = delete_after(head, 20); // delete node after 20
```

```
display(head);

return 0;
}
```

Output

Clear

```
Elements in the list:
64 68
```

```
Elements in the list:
10 64 68
```

```
Elements in the list:
10 64 68 20
```

```
Item deleted: 10
```

```
Elements in the list:
64 68 20
```

```
Item deleted: 20
```

```
Elements in the list:
64 68
No node found after 20
```

```
Elements in the list:
64 68
```

```
=== Code Execution Successful ===
```