1.Sum of all elements in linked list using recursion and without using recursion

Using recursion

```
#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));
newNode->data = data;
newNode->next = NULL;
return newNode;
}
int sumOfLinkedList(struct Node* head) {
If (head == NULL) {
return 0;
}
return head->data + sumOfLinkedList(head->next);
}
struct Node* createLinkedList(int* arr, int size) {
struct Node* head = NULL;
struct Node* tail = NULL;
for (int i = 0; i < size; i++) {
```

```
struct Node* newNode = createNode(arr[i]);
if (head == NULL) {
head = newNode;
tail = newNode;
} else {
tail->next = newNode;
tail = newNode;
}
}
return head;
}
void freeLinkedList(struct Node* head) {
struct Node* temp;
while (head != NULL) {
temp = head;
head = head->next;
free(temp);
}
}
int main() {
int arr[] = {1, 2, 3, 4, 5};
int size = sizeof(arr) / sizeof(arr[0]);
struct Node* head = createLinkedList(arr, size);
int sum = sumOfLinkedList(head);
```

```
printf("Sum of elements in the linked list: %d\n", sum);
freeLinkedList(head);
return 0;
}
```

Without using recursion

```
#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));
newNode->data = data;
newNode->next = NULL;
return newNode;
}
int sumOfLinkedList(struct Node* head) {
int sum = 0;
struct Node* current = head;
while (current != NULL) {
sum += current->data;
current current->next;
}
```

```
return sum;
}
struct Node* createLinkedList(int* arr, int size) {
struct Node* head = NULL;
struct Node* tail = NULL;
for (int i = 0; i < size; i++) {
struct Node* newNode = createNode(arr[i]);
if (head == NULL) {
head = newNode;
tail = newNode;
} else {
tail->next = newNode;
tail = newNode;
}
}
return head;
}
void freeLinkedList(struct Node* head) {
struct Node* temp;
while (head != NULL) {
temp = head;
head head->next;
free(temp);
}
```

```
}
int main() {
int arr[] = {1, 2, 3, 4, 5};
int size = sizeof(arr) / sizeof(arr[0]);
struct Node* head = createLinkedList(arr, size);
int sum = sumOfLinkedList(head);
printf("Sum of elements in the linked list: %d\n", sum);
freeLinkedList(head);
return 0;
}
```

2.counting no.of nodes in a linkedlist

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
int data;
struct Node *next;
};
void count(struct Node* head);
int main() {
struct Node head = (struct Node)malloc(sizeof(struct Node));
struct Node* first = (struct Node*)malloc(sizeof(struct Node));
head->next = first;
first->data = 10;
struct Node* second = (struct Node*)malloc(sizeof(struct Node));
second->data = 20;
first->next = second;
struct Node* third = (struct Node*)malloc(sizeof(struct Node));
third->data = 50;
second->next = third;
third->next = NULL;
count(head);
free(head);
free(first);
free(second);
free(third);
return 0;
}
void count(struct Node* p) {
int count = 0;
while(p != NULL) {
count++;
p = p->next;
}
printf("The count of nodes is: %d\n", count);
}
```

3. counting no.of nodes in a linkedlist using recursion

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

```
struct Node {
int data;
struct Node *next;
};
int count(struct Node* head);
int main() {
struct Node head = (struct Node)malloc(sizeof(struct Node));
struct Node* first = (struct Node*)malloc(sizeof(struct Node));
head->next = first;
first->data = 10;
struct Node* second = (struct Node*)malloc(sizeof(struct Node));
second->data = 20;
first->next = second;
struct Node* third = (struct Node*)malloc(sizeof(struct Node));
third->data = 50;
second->next = third;
third->next = NULL;
printf("The count of nodes is: %d\n", count(head));
free(head);
free(first);
free(second);
free(third);
return 0;
}
```

```
int count(struct Node* p) {
if (p == NULL) {
return 0;
}
return 1 + count(p->next);
}
```

4. finding out max element in a linkedlist

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
int data;
struct Node *next;
};
int findMax(struct Node* head);
int main() {
struct Node head = (struct Node)malloc(sizeof(struct Node));
struct Node* first = (struct Node*)malloc(sizeof(struct Node));
head->next = first;
first->data = 10;
struct Node* second = (struct Node*)malloc(sizeof(struct Node));
second->data = 20;
first->next = second;
struct Node* third = (struct Node*)malloc(sizeof(struct Node));
third->data = 50;
second->next = third;
third->next = NULL;
printf("The maximum element in the linked list is: %d\n", find Max(head));
free(head);
free(first);
free(second);
free(third);
return 0;
}
int findMax(struct Node* p) {
if (p == NULL) {
return -1;
}
int maxInRest = findMax(p->next);
if (maxInRest > p->data) {
return maxInRest;
} else {
return p->data;
}
}
```

5. searching for an element in the linkedlist

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
```

```
int data;
struct Node *next;
};
int search(struct Node* head, int target);
int main() {
struct Node head = (struct Node)malloc(sizeof(struct Node));
struct Node* first = (struct Node*)malloc(sizeof(struct Node));
head->next = first;
first->data = 10;
struct Node* second = (struct Node*)malloc(sizeof(struct Node));
second->data = 20;
first->next = second;
struct Node* third = (struct Node*)malloc(sizeof(struct Node));
third->data = 50;
second->next = third;
third->next = NULL;
int target = 20;
if (search(head, target)) {
printf("Element %d found in the linked list.\n", target);
} else {
printf("Element %d not found in the linked list.\n", target); }
free(head);
free(first);
free(second);
free(third);

return 0;

}

int search(struct Node* p, int target) {

if (p == NULL) {

return 0;

}

if (p->data == target) {

return 1; // Element found

}

return search(p->next, target);

}
```