

Search an element in a Linked List (Iterative and Recursive)

Write a function that searches a given key 'x' in a given singly linked list. The function should return true if x is present in linked list and false otherwise.

```
bool search(Node *head, int x)
```

For example, if the key to be searched is 15 and linked list is 14->21->11->30->10, then function should return false. If key to be searched is 14, then the function should return true.

Iterative Solution

- 1) Initialize a node pointer, current = head.
- 2) Do following while current is not NULL
 - a) current->key is equal to the key being searched return true.
 - b) current = current->next
- 3) Return false

Following is iterative implementation of above algorithm to search a given key.

```
// Iterative C++ program to search
// an element in linked list
#include <bits/stdc++.h>
using namespace std;

/* Link list node */
class Node
{
    public:
    int key;
    Node* next;
};

/* Given a reference (pointer to pointer) to the head
```

```

of a list and an int, push a new node on the front
of the list. */
void push(Node** head_ref, int new_key)
{
    /* allocate node */
    Node* new_node = new Node();

    /* put in the key */
    new_node->key = new_key;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Checks whether the value x is present in linked list */
bool search(Node* head, int x)
{
    Node* current = head; // Initialize current
    while (current != NULL)
    {
        if (current->key == x)
            return true;
        current = current->next;
    }
    return false;
}

/* Driver program to test count function*/
int main()
{
    /* Start with the empty list */
    Node* head = NULL;
    int x = 21;

    /* Use push() to construct below list

```

```

14->21->11->30->10 */
push(&head, 10);
push(&head, 30);
push(&head, 11);
push(&head, 21);
push(&head, 14);

search(head, 21)? cout<<"Yes" : cout<<"No";
return 0;
}

```

Output:

Yes

Recursive Solution

```

bool search(head, x)
1) If head is NULL, return false.
2) If head's key is same as x, return true;
3) Else return search(head->next, x)

```

Following is the recursive implementation of the above algorithm to search a given key.

```

// Recursive C++ program to search
// an element in linked list
#include <bits/stdc++.h>
using namespace std;

/* Link list node */
struct Node
{
    int key;
    struct Node* next;
};

/* Given a reference (pointer to pointer) to the head
of a list and an int, push a new node on the front

```

```

of the list. */
void push(struct Node** head_ref, int new_key)
{
    /* allocate node */
    struct Node* new_node =
        (struct Node*) malloc(sizeof(struct Node));

    /* put in the key */
    new_node->key = new_key;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Checks whether the value x is present in linked list */
bool search(struct Node* head, int x)
{
    // Base case
    if (head == NULL)
        return false;

    // If key is present in current node, return true
    if (head->key == x)
        return true;

    // Recur for remaining list
    return search(head->next, x);
}

/* Driver code*/int main()
{
    /* Start with the empty list */
    struct Node* head = NULL;
    int x = 21;

```

```
/* Use push() to construct below list
14->21->11->30->10 */
push(&head, 10);
push(&head, 30);
push(&head, 11);
push(&head, 21);
push(&head, 14);

search(head, 21)? cout << "Yes" : cout << "No";
return 0;
}
```

Output:

Yes