

isEmpty & isFull

Stack Operations Using Linked Lists

top

7 → 15 → 18 → 28 → NULL

7
15
18
28

1: isEmpty() :

```
Void isEmpty :  
    if (top == NULL) {  
        return 1;  
    }  
    else { return 0; }
```

2: isFull() :

```
struct Node * n  
    = (struct Node *) malloc (sizeof(Node))  
    if (n == NULL) {  
        return 1;  
    }  
    else {  
        return 0;  
    }
```

Algorithms & Data Structures > Stack Op LL

push

Stack Operations Using Linked Lists

1. isEmpty():

```
Void isEmpty() {
    if (top == NULL) {
        return 1;
    }
    else {
        return 0;
    }
}
```

2. isFull():

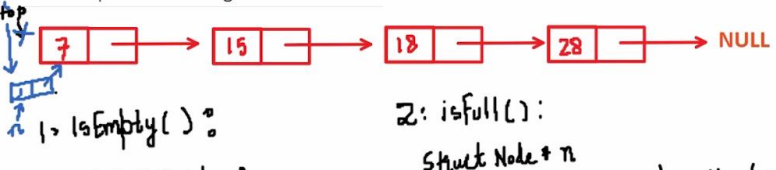
```
Struct Node * n
= (Struct Node *) malloc(sizeof(Node))
if (n == NULL) {
    return 1;
}
else {
    return 0;
}
```

3. Push(x): Inserting node @ index 0

```
Struct Node * n = (Struct Node *) malloc(sizeof(Node))
if (n == NULL) {
    printf("Stack overflow");
}
else {
    n->data = x;
    n->next = top;
    top = n;
}
```

pop

Stack Operations Using Linked Lists



```
graph LR
    top --> n1[7]
    n1 --> n2[15]
    n2 --> n3[18]
    n3 --> n4[28]
    n4 --> NULL
```

1. isEmpty():

```
Void isEmpty() {
    if (top == NULL) {
        return 1;
    }
    else { return 0; }
}
```

2. isFull():

```
struct Node * n = (struct Node *) malloc(sizeof(Node));
if (n == NULL) {
    return 1;
} else {
    return 0;
}
```

3. Push(x): Inserting node @ index 0

```
struct Node * n = (struct Node *) malloc(sizeof(Node));
if (n == NULL) {
    printf("Stack overflow");
} else {
    n->data = x;
    n->next = top;
    top = n;
}
```

4. Pop:

```
if (isEmpty) {
    printf("Stack underflow");
} else {
    struct Node * n = top;
    top = top->next;
    int x = n->data;
    free(n);
    return x;
}
```