

Implementation of stack

1- Pointer imp.

2- Array imp.

```

struct Node;
typedef int struct Node * Ptr;
typedef Ptr stack;
struct Node {
    int Data;
    Ptr next;
};
    
```

stack createStack() {

stack s;

s = (stack) malloc(sizeof(struct Node));

if (s == NULL) {

printf("Error: out of memory");

return -1;

}

s->next = NULL;

return s;

}

void push(stack s, int x) {

Ptr temp;

temp = (Ptr) malloc(sizeof(struct Node));

if (temp != NULL) {

temp->Data = x;

temp->next = s->next;

s->next = temp;

}

else

}

void pop(stack s) {

Ptr temp;

```

if( isEmpty(s) ) {
    printf( "Error: do not pop from empty stack" );
} else {
    temp = s->next;
    s->next = s->next->next;
    free(temp);
}
}

```

```

int isEmpty(stack s) {
    return (s->next == NULL);
}

```

```

int Top(stack s) {
    if(! isEmpty(s))
        return s->next->Data;
    else
        return -1;
}

```

```

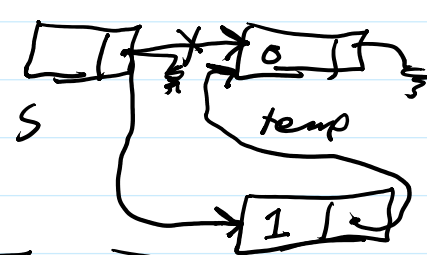
void main() {

```

```

    stack s = createStack();
    int i = 0;
    while(i < 5) {
        push(i);
        ++i;
    }

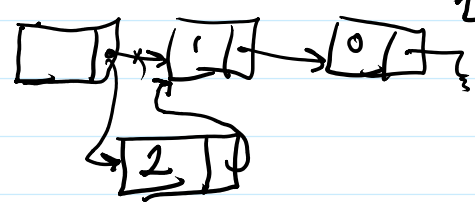
```



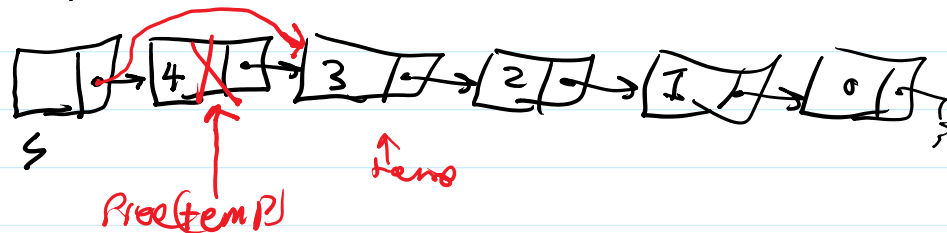
```

pop(s);
pop(s);

```



LIFO



```

s->next = s->next->next;

```

ptr p;

Preempt

s->next = s->next->next;

```
void makeEmpty(stack s) {  
    while(!isEmpty(s))  
        pop(s);  
}
```

ptr p;

p = s;

```
while (p != NULL) {  
    printf("%d", p->data);  
    p = p->next;  
}
```

Array Imp. of stack.

LI FO