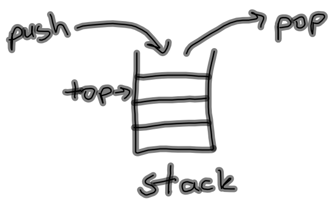
STACKS

A stack is a container of objects that are inserted and removed according to the last-in-first out LIFO principle

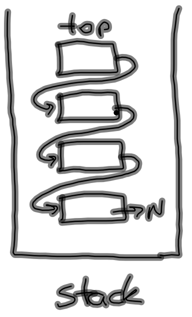


push ( ): Adds item to top
pop ( ): removes the item from top

real life examples
pile of dinner plates
Batteries of a light

Implementation of Stacks

1. Linked List implementation



```
typedef struct{
    struct node *top;
    int cnt;
}stack;

#define STACK-SIZE 10
```

Operations

a) Initialization

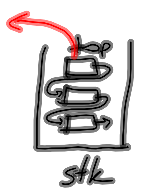
```
void initialize( stack *stk){
        stk->top= NULL;
        stk->cnt= 0;
}
```

b) Push operator

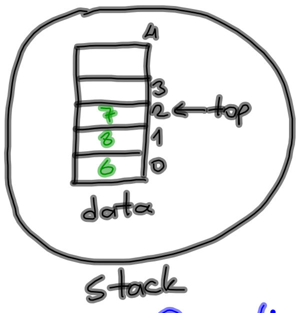
```
void push( stack *stk ,int c){
    if (stk->cnt== STACK_SIZE)
         printf ("stack is full");
    else{
        struct node *temp= ... malloc....
        temp->data=c;
        temp->next= top;
        top= temp;
        stk->cnt++;
    }
}
```

b) Pop operator

```
int popstack *stk){
    if(stk->cnt==0){
        printf("Stack is empty");
        return -100;
    }else{
        int x= stk->top->data;
        struct node *temp=top;
        top=top->next;
        free(temp);
        stk->cnt--;
        return x;
    }
}
```



2-) Array Implementation



```
typedef struct{
    int data[STACK-SIZE];
    int top;
}stack;
```

Operations

a) Reseting

```
void reset(stack *stk){
    stk->top=-1;
}
```

b) Push operator

```
void push(stack *stk, int c){
    if(stk->top == STACK_SIZE-1)
        printf("stack is full");
    else{
        stk->top++;
        stk->data[stk->top]=c;
    }
}
```

stk->data[++stk->top]=c;

c) Pop operator

```
int pop(stack *stk){
    if( stk->top==-1 ){
        printf ("Stack is empty");
        return -100;
    }else{
        int x= stk->data[stk->top];
        stk->top--;
        return x;
    }
}
```

```
return stk->data[stk->top--];
```

STACK_SIZE=3 kabul edin

```
main ( ) {
    stack n;
    reset ( &n);
    push ( &n, 4);
    push ( &n,14);
    push ( &n, 41);
    push ( &n, 6);
}
```

```
printf("%d\t", pop(&n));
printf("%d\t", pop(&n));
printf("%d\t", pop(&n));
printf("%d\t", pop(&n));
```

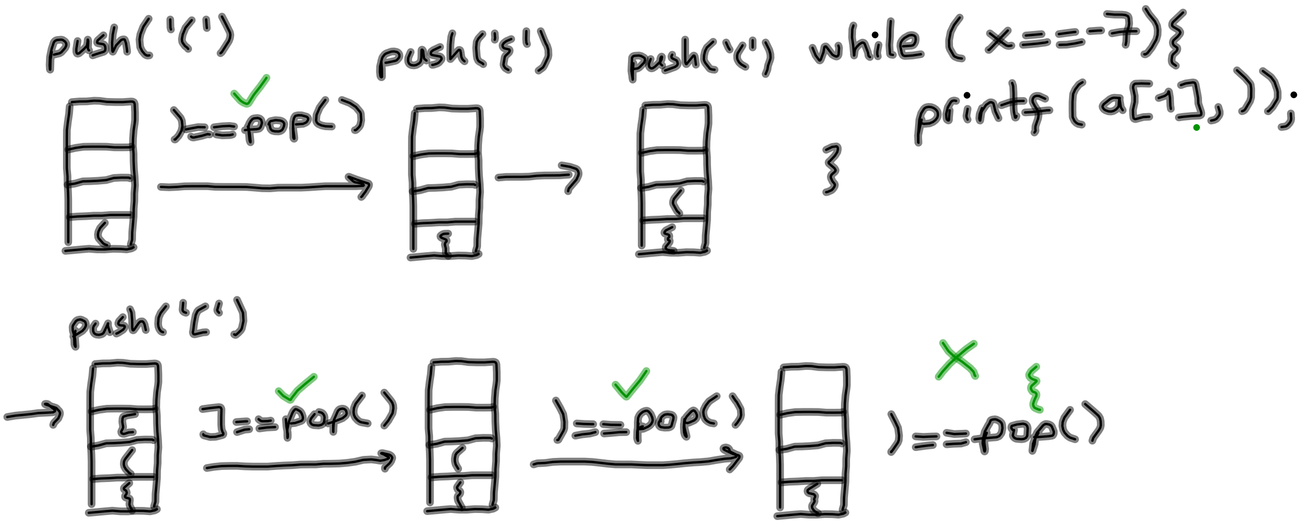
Application of Stacks in Computers

- In recursive function
- Compilers check programs for syntax errors, but Just parenthesis.

```
while ( x==-7){
      printf ( a[1],));
}
```

- ( ) bracets
- { } braces or curly bracets
- [ ] square bracets

for (, {, [we perform push operator
for), },] we " pop " and compare their type



push('(')
✓
)==pop()
push('{')
push('(')
while (x==-7){
printf (a[1],));
}
push('[')
✓
]==pop()
✓
)==pop()
✗
{
)==pop()