October 7

Stack - linear Data structure which follows a particular order in which operations are performed.

1) LIFO (Last In first out)

2) FILO (first in first out)

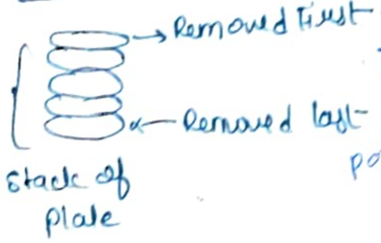
Operations in stack.

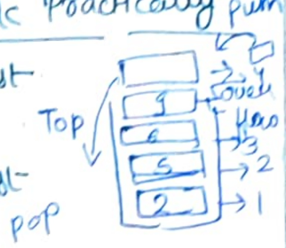
(a) push - add an item + If stack is full → overflow condn

(b) pop - removes an item (items are popped in reverse order in which they are pushed) + If stack is empty → under flow condition (d) Isempty / Is full

(c) Peek - Getting Top most item.

Understanding stack Practically





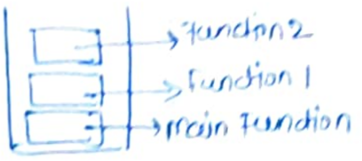
Implementation → using array
→ using linked list

structure to represent a stack

```
struct stack
  { int top;
    unsigned capacity;
    int * array;
  }
```

Applications of stack

- Used in function calls
- Infix to postfix conversion
- Parenthesis matching & more



main function will call each function.

Stack as abstract Data type

→ In order to create a stack we need a pointer to the topmost element along with other elements which are stored inside the stack.

Stack using Array Data structure

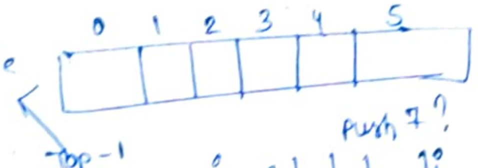
→ Fixed size array creation

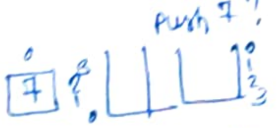
→ Top Element

```
Struct Stack {
        int size;
        int top;
      } int arr;
```

```
Struct stack s;
 s. size = 80;
 s. top  = -1;
 s. arr = (int *) malloc (s.size * sizeof int)
              malloc (s.size * size of
                      (int));
```

Example





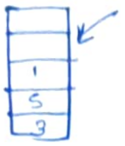
Now, Top = 0 ..
↓
Denoting Till when stack is filled.

Above we are performing push 6 pull operation on stack using array.

October 8

ALGORITHMS ON STACK

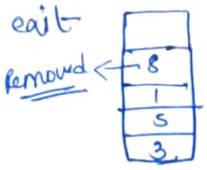
PUSH



```
Step1 - Start
Step2 - If Top = max
        print overflow & exit
Step3 - Read Data
Step4 - Top = Top+1
Step5 - Stack [Top] = Data
Step6 - Exit
```

POP ALGORITHM

```
Step1 - Start
Step2 - If Top = Null
        print underflow & exit
Step3 - stack [Top] = Null
Step4 - Top = Top-1
Step5 - Exit
```



STACK NOTIFICATION

1) INFIX - when operator comes between operands eg:- A+B-C*D

2) PREFIX - [+ AB] when operator Is putted before operand.

3) POSTFIX - AB+ when operator is putted after operand.

ALGORITHM FOR CONVERTING FROM INFIX TO POSTFIX :-

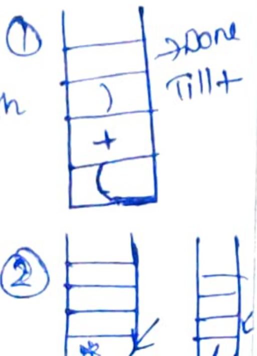
S = ((A+B)*C/D)

Final R = AB+C*D/

① Step 1 - start

Step 2 - Push "(" on to stack 2add ")" to end of stack

Step 3 - Scan 'S' Left to Right two repeat step 3 to 6, untill S is empty.



^ - Highest precedence ①

* / - high level precedence ②

+ - - lower than 1 and 2

Step4 - If an operator ⊕ is encountered

(a) Add operator on-to-the stack

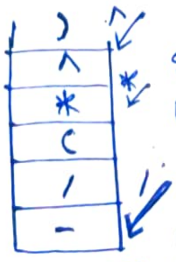
(b) If another operator Exist 2 New operator has less or equal precedence then pop the stack 2 add to R push new operator onto top of stack

R = AB+C*D/

ALGORITHM FOR CONVERSION FROM INFIX TO POSTFIX

convert to post : A-B/(C*D^E)

```
Step1 - Start
Step2 - Push "(" onto Stack & add ")" to
        end of stack.
Step3 - Scan 'S' Left to Right two repeat
        Step 3 to 6 untill S is empty
Step4 - If operator is encountered
    (a) Add operator on to the stack
    (b) If another operator Exist 2
        New operator has less or equal
        precedence pop the stack to add to
        R push new operator onto top
```



S = A-B/(C*D^E))

R = ABCDE^*/-

∴ / has high precedence than (-)

① →

② (parantheses & multiply

^

* /

+ -

October 8

ack - linear Data structure
hich follows a particular order in
hich operations are performed.
) LIFO (last In first out)
) FILO (first in first out)

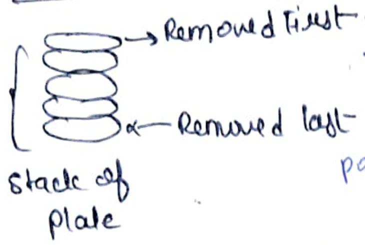
Operations in stack .

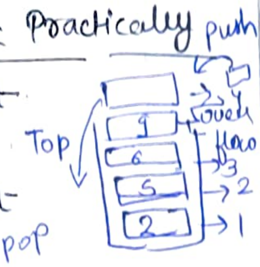
a) push - add an item + If stack is full → overflow condn

b) pop - removes an item (items are popped in reverse order in which they are pushed) + If stack is empty → underflow condition (d) Isempty / Is full

(c) Peek - Getting Top most item.

Understanding stack Practically





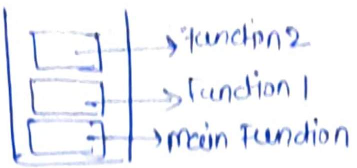
Implementation → using array
→ using linked list

structure to represent a stack

```
struct Stack
{ int top;
  unsigned capacity;
  int * array;
}
```

Applications of stack

- used in Function calls
- Infix to postfix conversion
- Parenthesis matching & more



main function will call each function.

Stack as abstract Data type

→ In order to create a stack we need a pointer to the topmost element along with other elements which are stored inside the stack.

Stack using Array Data structure

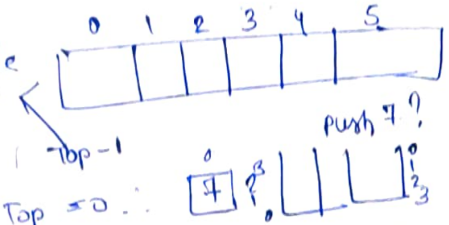
→ Fixed size array creation

→ Top element

```
Struct Stack {
    int size;
    int top;
    int arr;
}
```

```
Struct stack s;
  s. size = 80;
  s. top = -13
  s. arr = (int *) malloc (s.size * size of int)
           malloc (s.size * size of (int));
```

Example



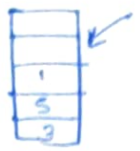
Now, Top = 0 .
↓
Denotify Till when stack is filled.

Above we are performing push & pull operation on stack using array.

October 8

ALGORITHMS ON STACK

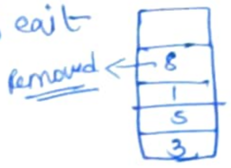
PUSH



Step 1 - Start
Step 2 - If Top = max
print overflow & exit
Step 3 - Read Data
Step 4 - Top = Top + 1
Step 5 - Stack [Top] = Data
Step 6 - Exit

Pop ALGORITHM

Step 1 - Start
Step 2 - If Top = Null
print underflow & exit
Step 3 - stack [Top] = Null
Step 4 - Top = Top - 1
Step 5 - Exit



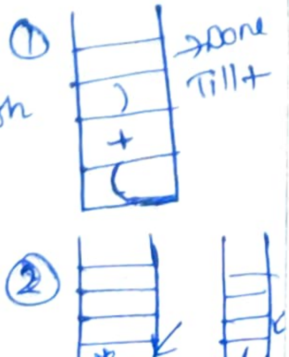
STACK NOTIFICATION

1) INFIX - when operator comes between operands eg:- A+B-C*D
2) PREFIX - [+ AB] when operator is putted before operand.
3) POSTFIX - AB+ when operator is putted after operand.

ALGORITHM FOR CONVERTING FROM INFIX TO POSTFIX :-

S = ((A+B)*C/D)
Final R = AB+C*D/

① Step 1 - start
Step 2 - Push "(" on to stack 2 add ")" to end of stack
Step 3 - Scan 'S' left to Right two repeat step 3 to 6, until S is empty.



^ - Highest precedence ①
* / - high level precedence ②
+ - - lower than 1 and 2

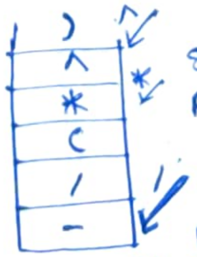
Step 4 - If an operator ⊕ is encountered
(a) Add operator on-to-the stack
(b) If another operator Exist 2 New operator has less or equal precedence then pop the stack 2 add to R push new operator onto top of stack

R = AB+C*D/

ALGORITHM FOR CONVERSION FROM INFIX TO POSTFIX

convert to post : A-B/(C*D^E)

Step 1 - start
Step 2 - Push "(" onto stack 2 add ")" to end of stack.
Step 3 - Scan 'S' left to Right two repeat Step 3 to 6 until S is empty
Step 4 - If operator is encountered
ⓐ Add operator on to the stack
ⓑ If another operator Exist 2 New operator has less or equal precedence pop the stack to add to R push new operator onto top



S = A-B/(C*D^E))
R = ABCDE^*/-

∴ / has high precedence than (-)

① →
② (parantheses & multiply

^
* /
+ -