01/01/2024

* Stack

```
#define size 5
top = -1, stack[size];

void push(int value){
    if (top == size-1){
        print(overflow);}
    else{
        top++;
        stack[top] = value;
    }

void pop(){
    if (top == -1){
        print(underflow);}
    else{
        int val = stack[top];
        top--;
        print(val);
    }

void display(){
    if (top == -1){
        print(stack is empty);}
    else{
        for(int i = top; i >= 0; i--){
            print(stack[i]);
        }
    }
```

Date: 09/01/2024

* Infix to postfix conversion

```
void push (char symbol){
    top++;
    stack[top] = symbol;

char pop (){
    char val = stack[top];
    top--;
    return val;
}

int precidence (char symbol){
    int point;
    switch (symbol){
        case '^':
            point = 3;
            break;
        case '*':
        case '/':
            point = 2;
            break;
        case '+':
        case '-':
            point = 1;
            break;
        case '(':
            point = 0;
            break;
        case '#':
            point = -1
            break;
    }
```

```
void conversion() {
    int index;
    int length = strlen(infix);
    push('#');
    while (length > index) {
        char val = infix[index];
        switch (val) {
            case '(':
                push(val);
                break;
            case ')':
                val = pop()
                while (val != '(') {
                    postfix[pos] = val;
                    pos++;
                    val = pop()
                };
            case '*':
            case '+':
            case '-':
            case '/':
                while (precidence[stack[top]) > precidence(val)
                    val = pop();
                    postfix[pos] = val;
                    pos++;
                };
            default:
                postfix[pos] = val;
                pos++;
        }
        index++
    }
```

11/12/24

```
    while (top > 0) {
        val = pop();
        postfix [pos] = val;
        pos++;
    }
}
```

O/p:

Enter the infix expression : a*b+c*d-e

infix expression : a*b+c*d-e

postfix expression : ab*cd*+e-

Output for stack program

*** MENU ****

Enter 1 : to push

Enter 2 : to pop

Enter 3 : to display

Enter 4 : to exit

1

Enter the element to push: 32

Insertion Successful....

2

poped element: 32

3

Stack is empty.

46

Invalid number. try again

4 (exists)

Sen 11/12/24

```
    while (top > 0) {
        val = pop();
        postfix [pos] = val;
        pos++;
    }
}
```

O/p:

Enter the infix expression : a*b+c*d-e

infix expression : a*b+c*d-e

postfix expression : ab*cd*+e-

Output for stack program

*** MENU ****

Enter 1 : to push

Enter 2 : to pop

Enter 3 : to display

Enter 4 : to exit

1

Enter the element to push: 32

Insertion Successful....

2

poped element : 32

3

Stack is empty

46

Invalid number, try again

4 (exists)

Sen 11/12/24