1) Write a program to simulate the working of stack using an array with the following:
   a) Push
   b) pop
   c) Display.
The program should print appropriate messages for stack overflow, stack underflow

→ define size 5;
int stack[size]
int top = -1;

Push:

void push() {
int num
printf("Enter the value");
scanf("%d", &num);
if (top == size-1) {
printf("over flow").
}
else.
top++
stack[top] = element
print → "pushed"

pop:

void pop() {
int num;
if (top == -1) {
print → (underflow)
else:

```
num = stack[top];
top--;
printf("pop-ed");
```

```
display(){
void display(){
for(int i = top; i>=0; i--){
printf("%d", stack[i]);
```

2.) WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operands +, -, *, /.

```
int operator(char x){
return (x=='*' || a=='/' || a=='+' || a=='-');
}

int precedence(char s){
if(s=='*' || s=='/'){
return 2;
}
else if(s=='+' || s=='-'){
return 1;
}
else{
return 0;
}

void infixtopostfix(char *infix){
int i=0, j=0;
char postfix[MAX_SIZE];
while(infix[i] != '\0'){
if(!operator(infix[i])){
```

```
postfix[j] = infix[i];
  j++;
  i++;
}else
  while (top >= 0 && preedence(stack[top]
  >= preedence(infix[i])){
      postfix[j] = pop();
      j++;
  }
  push(infix[i);
  i++
}
}
while(top >= 0){
  postfix[j] = pop();
  j++
}
postfix[j] = '\0'
```

Input → infix
print → postfix.

Sw
1/1/24

OUTPUT:

Enter infix: a+*b/d

ab*d/+.