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

```
#include <stdio.h>
#include <ctype.h>
# include <string.h>
# define MAX 100
char stack[MAX];
int top =-1;
void push (char c)
 {
     if (top==MAX-1){
         printf("Stack Overflow\n");
         return;
     }
     stack [++top] =c;
 }
Char pop(){
   if (top==-1){
        printf ("Stack Underflow\n");
        return-1;
     }
   return stack[top--];
}
char peek (){
     if (top==-1){
          printf ("Stack Underflow\n");
          return-1;
     }
     return stack [ top ];
}
```

```
int precedence(char op){
    switch(op){
        case'+':
        case'-':
            return 1;
        case'*':
        case'/':
            return 2;
        case'^':
            return 3;
        case'(':
            return 0;
        default:
            return -1;
}
  ~~retur~~

// 0 = Left-to-Right , 1 = Right-to-Left
int ~~pre~~ associativity(char op){
    if(op=='^'){
        return 1;
    }
    return 0;
}
void infixtopostfix(char infix[], char postfix[]) {
  ~~int~~ i,k=0;
   char c;
  for (i=0;infix[i]!='\0';i++) {
      c= infix[i];
      if(isalnum(c)){
           postfix[k++]=c;
      }
      else if (c=='('){
           push(c);
      }
      else if (c==')') {
            while (peek() != '(') {
                        postfix[k++] = pop();

            }
            pop();

        }
        else {
          while (top != -1 && ((precedence(peek()) >
                                  precedence(c)) ||
                                 (precedence(peek()) ==
                                 precedence(c) &&
                                 accociativity(c) == 0))) {

                  postfix[k++] = pop();
            }
             push(c);
          }
   }
   while (top != -1) {
          postfix[k++] = pop();
   }
   postfix[k] = '\0';

}
int main() {
    char infix[MAX], postfix[MAX];
    Printf ("Enter a valid parenthesized infix expression:");
    Scanf ("%s", infix);
    infixtopostfix (infix, postfix);
    Printf ("Postfix Expression: %s\n", postfix);
}   return0;
```

[illegible] 13/10/24

O/P:-

Enter a valid parenthesized infix expression:-
(A+(B*C-(D/E^F)*G)*H)

- Postfix Expression: ABC*DEF^/G*-H*+

B|
13/10/25 lab pgm-3

a) WAP to simulate the working of queue of integers using an array. Provide the following operations: Insert, Delete, Display. The pgm should print appropri-ate messages for queue. empty & overflow conditions

- code