```
/* Program For Conversion Of Infix Expression To Postfix Expression */ #include #define MAX
100 // Stack for operators char stack[MAX]; int top = -1; // Function to push an element onto the
stack void push(char ch) { if (top >= MAX - 1) printf("Stack Overflow\n"); else stack[++top] = ch; }
// Function to pop an element from the stack char pop() { if (top < 0) { printf("Stack
Underflow\n"); return -1; } else { return stack[top--]; } } // Function to check the precedence of
operators int precedence(char operator) { switch (operator) { case '+': case '-': return 1; case '*':
case '/': return 2; case '^': return 3; default: return 0; } } int main() { char infix[MAX], postfix[MAX],
ch, temp ; int i = 0, j = 0; printf("Enter an infix expression: "); scanf("%s",&infix); // Take input for
infix expression while ((ch = infix[i++]) != '\0') { if (isalnum(ch)) postfix[j++] = ch; // If the character
is an operand, add it to postfix else if (ch == '(') push(ch); else if (ch == ')') { // Pop until '(' is
found while ((temp = pop()) != '(') postfix[j++] = temp; } else { // Operator encountered while (top
!= -1 && precedence(stack[top]) >= precedence(ch)) postfix[j++] = pop(); push(ch); } } // Pop the
remaining operators while (top != -1) postfix[j++] = pop(); postfix[j] = '\0'; // Null-terminate the
postfix expression printf("Postfix expression: %s\n", postfix); return 0; }
```