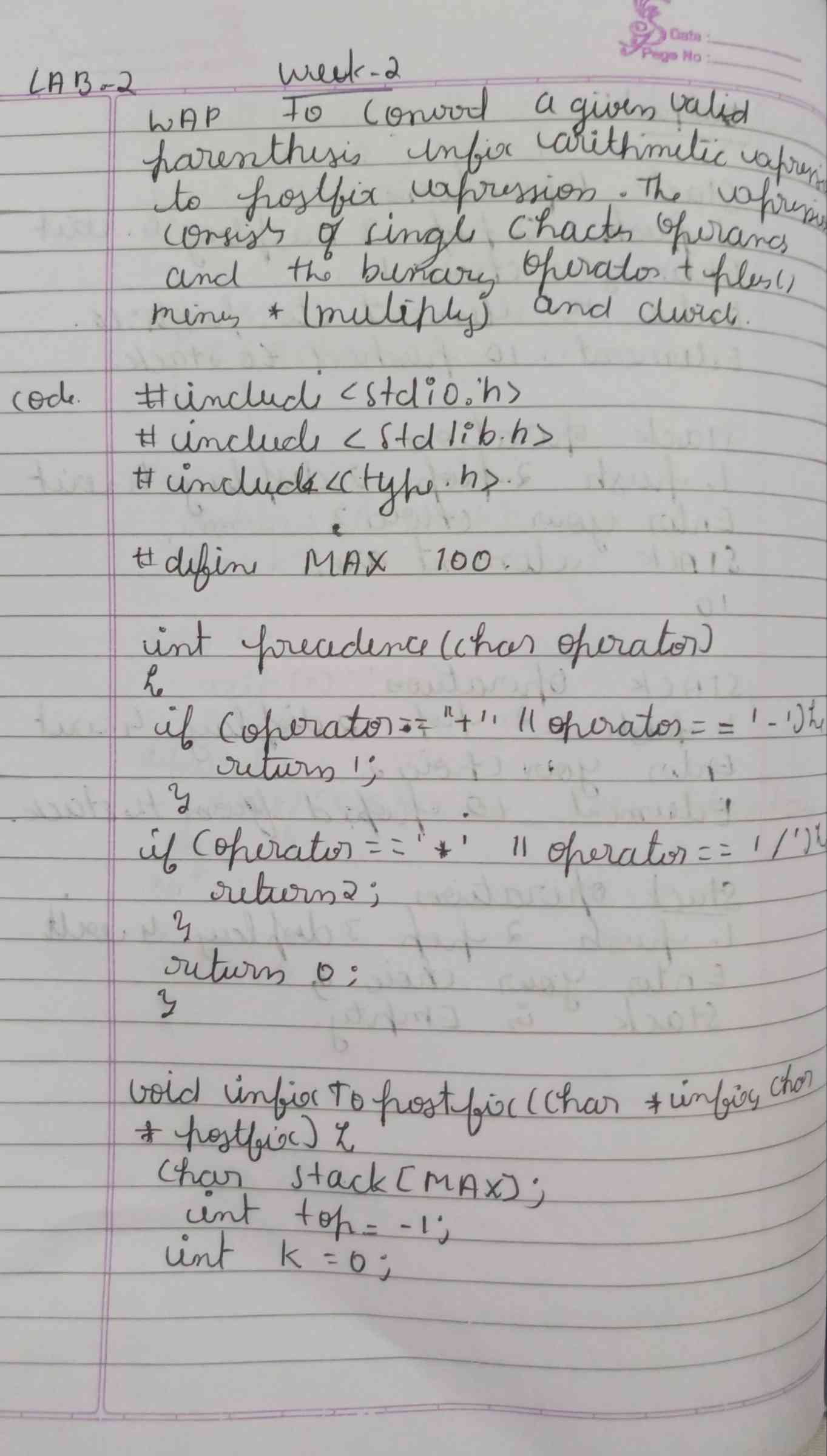
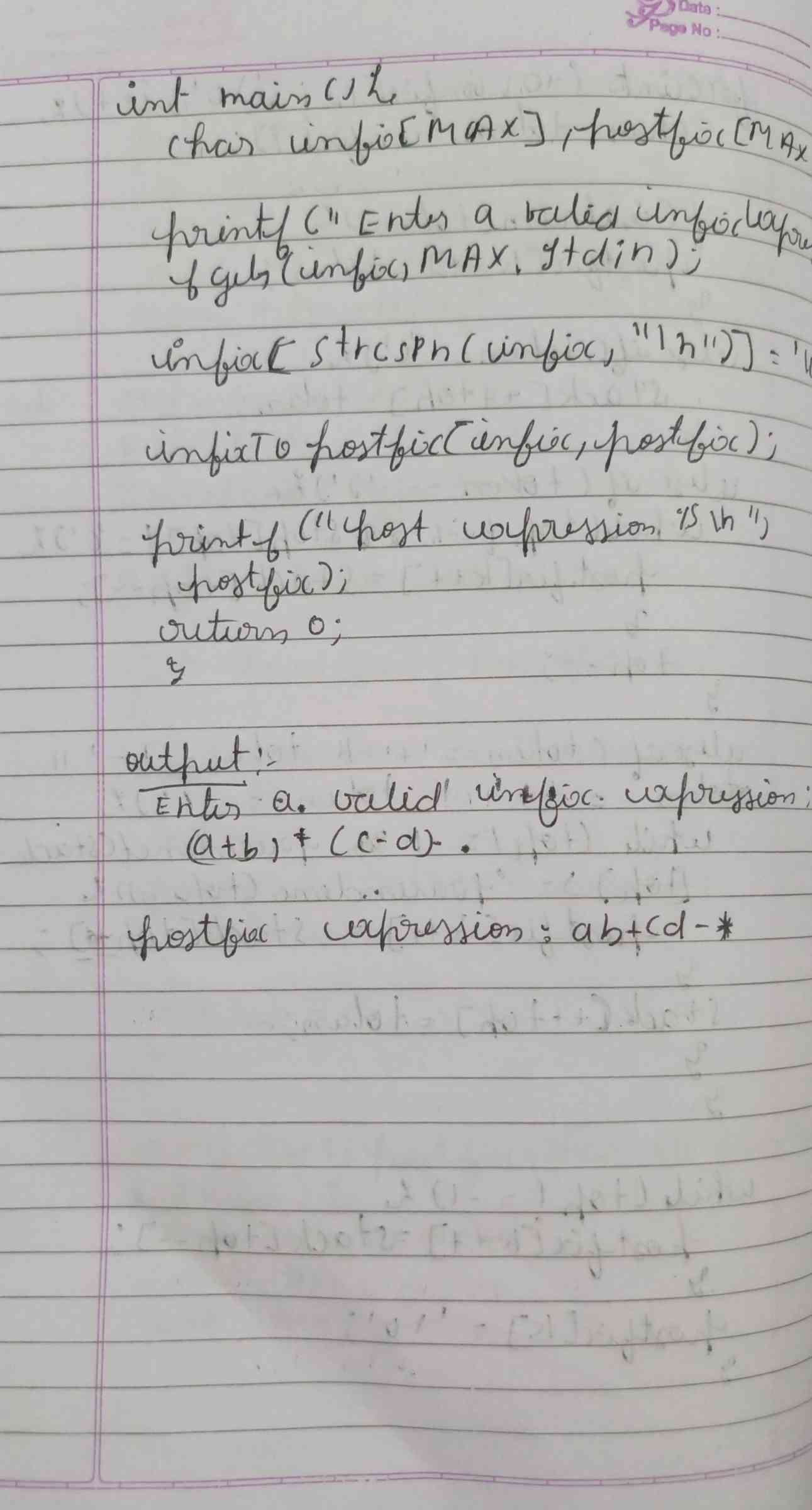
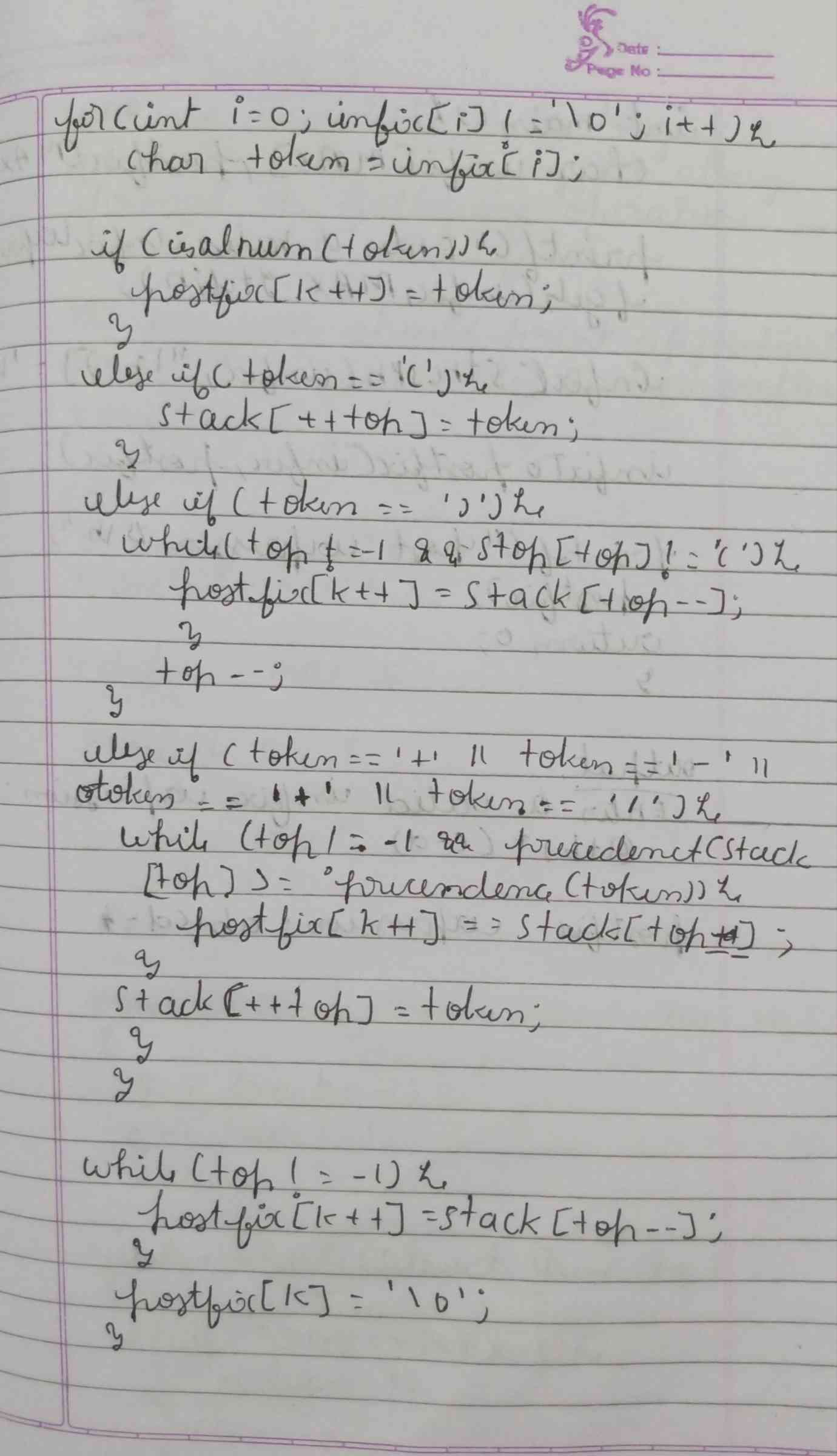
**LABORATORY PROGRAM – 2**

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), \* (multiply) and / (divide)

**OBSERVATION :**

****

****

**CODE :**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#define MAX 100**

**int prec(char c) {**

**if (c == '^')**

**return 3;**

**else if (c == '/' || c == '\*')**

**return 2;**

**else if (c == '+' || c == '-')**

**return 1;**

**else**

**return -1;**

**}**

**char associativity(char c) {**

**if (c == '^')**

**return 'R';**

**return 'L';**

**}**

**void infixToPostfix(const char \*s) {**

**char result[MAX];**

**char stack[MAX];**

**int resultIndex = 0;**

**int stackIndex = -1;**

**int len = strlen(s);**

**for (int i = 0; i < len; i++) {**

**char c = s[i];**

**if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9')) {**

**result[resultIndex++] = c;**

**}**

**else if (c == '(') {**

**stack[++stackIndex] = c;**

**}**

**else if (c == ')') {**

**while (stackIndex >= 0 && stack[stackIndex] != '(') {**

**result[resultIndex++] = stack[stackIndex--];**

**}**

**stackIndex--;**

**}**

**else {**

**while (stackIndex >= 0 &&**

**(prec(c) < prec(stack[stackIndex]) ||**

**(prec(c) == prec(stack[stackIndex]) && associativity(c) == 'L'))) {**

**result[resultIndex++] = stack[stackIndex--];**

**}**

**stack[++stackIndex] = c;**

**}**

**}**

**while (stackIndex >= 0) {**

**result[resultIndex++] = stack[stackIndex--];**

**}**

**result[resultIndex] = '\0';**

**printf("Postfix expression: %s\n", result);**

**}**

**int main() {**

**char exp[MAX];**

**printf("Enter an infix expression: ");**

**fgets(exp, MAX, stdin);**

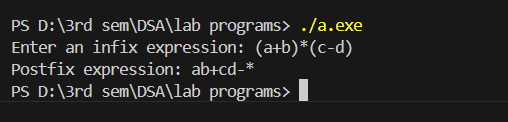
**exp[strcspn(exp, "\n")] = 0;**

**infixToPostfix(exp);**

**return 0;**

**}**

**OUTPUT :**

****