Q2(a)_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)

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define MAX 50

char stack[MAX];
int top = -1;

void push(char c) {
   stack[++top] = c;
}

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

int priority(char c) {
   if (c == '*' || c == '/')
      return 2;
   if (c == '+' || c == '-')
      return 1;
   return 0;
}
```

```
void infixToPostfix(char infix[], char postfix[]) {
  int i, j = 0;
    char item, x;

    for (i = 0; infix[i] != '\0'; i++) {
        item = infix[i];

        if (isalnum(item)) {
            postfix[j++] = item;    // operand → directly to postfix
        }
        else if (item == '(') {
            push(item);
        }
        else if (item == ')') {
            while (stack[top] != '('
                postfix[j++] = pop();
            pop(); // remove '('
        }
        else { // operator
            while (top != -1 && priority(stack[top]) >= priority(item))
                postfix[j++] = pop();
            push(item);
        }
    }

    while (top != -1)
        postfix[j++] = pop();
```

```
    postfix[j] = '\0';
}
int main() {
    char infix[MAX], postfix[MAX];

    printf("Enter a valid infix expression: ");
    scanf("%s", infix);

    infixToPostfix(infix, postfix);

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

    return 0;
}
```