/// algorithm after realizing stack for postfix not needed (stack and array are basically the same in C but with the difference that stack operations cannot be performed on ordinary array. Hence, in this program it saves the huge code for stack of strings and operations on it. ///

/// are ‘(‘ and ‘)’ compulsory at start and end of entered infix before applying this algorithm?? ///

1. Declare 1 stack of characters: oprtorstk
2. Declare a string for storing postfix expression: postfix
3. Enclose infix string in ‘(‘ and ‘)’ if not already enclosed (but instead of checking this, directly add a new set to every input expression, irrespective of whether already present or not).
4. Read each character of infix string one-by-one from start to end.
5. If chr==’(‘ :
6. Push chr in oprtorstk.

Else if chr==operand :

1. Append chr to postfix.

Else if chr==operator :

If oprtorstk->top==-1 : Push chr in oprtorstk.

Else :

1. If chr == ’$’ :
2. Push chr in oprtorstk.
3. Else If chr == ‘/’ || ‘\*’ :
4. If peek(oprtorstk) == ‘+’ || ’-‘|| ’(‘ :
5. Push chr in oprtorstk.
6. Else If peek(oprtorstk) == ‘\*’ || ‘/’ || ‘$’ :
7. While (oprtorstk->top >=0 && oprtorstk->top == ‘\*’ || ‘/’ || ‘$’ ) :

Pop oprtorstk and append to postfix.

1. Push chr in oprtorstk.
2. Else If chr == ‘+’ || ‘-‘ :
3. While (oprtorstk->top >=0 && peek(oprtorstk) != ‘(‘ ) :

Pop oprtorstk and append to postfix.

1. Push chr in oprtorstk.

Else if chr == ’)’ :

1. While (peek(oprtorstk) != ‘(‘ )

Pop oprtorstk and append to postfix.

1. Pop oprtorstk. (to remove ‘(‘).

Else :

Print “Unknown token in entered infix expression.”

1. Equivalent postfix expression = postfix

//// First attempt : A redundant stack ////

1. Declare 2 stacks:
2. Stack containing strings: solstk
3. Stack containing characters: oprtorstk
4. Enclose infix string in ‘(‘ and ‘)’ if not already enclosed.
5. Read each character of infix string one-by-one from start to end.
6. If chr==’(‘ :
7. Push chr in oprtorstk.
8. solstk->top++ (so infix string needs to start and end with ‘(‘ and ‘)’ respectively)

Else if chr==operand :

1. Append chr to newly topmost element of solstk.

Else if chr==operator :

If oprtorstk->top==-1 : Push chr in oprtorstk.

Else :

1. If chr == ’$’ :
2. Push chr in oprtorstk.
3. Else If chr == ‘/’ || ‘\*’ :
4. If peek(oprtorstk) == ‘+’ || ’-‘|| ’(‘ :
5. Push chr in oprtorstk.
6. Else If peek(oprtorstk) == ‘\*’ || ‘/’ || ‘$’ :
7. While (oprtorstk->top >=0 && oprtorstk->top == ‘\*’ || ‘/’ || ‘$’ ) :

Pop oprtorstk and append to solstk->top.

1. Push chr in oprtorstk.
2. Else If chr == ‘+’ || ‘-‘ :
3. While (oprtorstk->top >=0 && peek(oprtorstk) != ‘(‘ ) :

Pop oprtorstk and append to newly topmost element of solstk.

1. Push chr in oprtorstk.

Else if chr == ’)’ :

1. While (peek(oprtorstk) != ‘(‘ )

Pop oprtorstk and append to newly topmost element of solstk.

1. Pop oprtorstk. (to remove ‘(‘).
2. If solstk->top >= 1 :

Pop solstk and append to newly topmost element of solstk.

Else :

Print “Unknown token in entered infix expression.”

1. Postfix expression = solstk[0]