PRAKTIKUM STRUKTUR DATA

INF11198

2401020010 – FACHREZI BACHRI

1. Berikut source code yang sudah saya lengkapi soucode ini berguna untuk konversi rumus infix ke postfix. Saya hanya lengkapi kekurangan source code yang di berikan tanpa merubah yang sudah ada.

   Sourch code:

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

#define MAX 100

// Stack structure
typedef struct {
  int top;
  char items[MAX];
} Stack;

void push(Stack *s, char c){
  if (s->top == MAX - 1){
    printf("Stack Overflow\n");
    return;
  }
  s->items[++(s->top)] = c;
}
```

```
char pop(Stack *s){
  if(s->top == MAX - 1){
    printf("Stack Underflow\n");
    return '\0';
  }
  return s->items[(s->top)--];
}

char peek(Stack *s){
  return s->items[s->top];
}

int precedence(char op){
  if(op == '^') return 3;
  if(op == '*' || op == '/') return 2;
  if(op == '+' || op == '-') return 1;
  return 0;
}

int isOperator(char c){
  return c == '+' || c == '-' || c == '*' || c == '/' || c ==
'^';
}

void infixToPostfix(char *infix, char *postfix){
  Stack s;
  s.top = -1;
  int i = 0, j = 0;

  while(infix[i] != '\0'){
    if(isalnum(infix[i])){
```

```
        postfix[j++] = infix[i]; // Add operand to postfix
    } else if(infix[i] == '('){
        push(&s, infix[i]);
    } else if(infix[i] == ')'){
        while(s.top != -1 && peek(&s) != '('){
            postfix[j++] = pop(&s);
        }
        pop(&s); // Remove '('
    } else if(isOperator(infix[i])){
        while(s.top != -1 && precedence(peek(&s)) >=
precedence(infix[i])){
          postfix[j++] = pop(&s);
        }
        push(&s, infix[i]);
    }
    i++;
  }

  while(s.top != -1){
    postfix[j++] = pop(&s);
  }

  postfix[j] = '\0'; // Null-terminate the postfix expression
}



int main(){
  char infix[MAX], postfix[MAX];

  printf("Enter an infix expression: ");
  scanf("%s", infix);
```

```
    infixToPostfix(infix, postfix);

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

    return 0;
}
```

2. Berikut source code yang sudah saya lengkapi soucode ini berguna untuk konversi rumus infix ke prefix. Saya hanya lengkapi kekurangan source code yang di berikan tanpa merubah yang sudah ada.

   Sourch code:

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

#define MAX 100

// Stack structure
typedef struct {
  int top;
  char arr[MAX];
} Stack;

void push(Stack *stack, char c){
  stack->arr[++stack->top] = c;
}

char pop(Stack *stack){
  return stack->arr[stack->top--];
}

char peek(Stack *stack){
  return stack->arr[stack->top];
}
```

```
char isEmpty(Stack *stack){
  return stack->top == -1;
}

// Utility functions
int precedence(char op){
  if(op == '+' || op == '-') return 1;
  if(op == '*' || op == '/') return 2;
  if(op == '^') return 3;
  return 0;
}

int isOperator(char c){
  return c == '+' || c == '-' || c == '*' || c == '/' || c == '^';
}

// Reverse a string
void reverse(char *exp){
  int len = strlen(exp);
  for(int i = 0; i < len / 2; i++){
    char temp = exp[i];
    exp[i] = exp[len - i - 1];
    exp[len - i - 1] = temp;
  }
}

// Convert infix to prefix
void infixToPrefix(char *infix, char *prefix){
  Stack stack;
  stack.top = -1;

  reverse(infix); // Reverse the infix expression
```

```
  int j = 0;
  for(int i = 0; infix[i] != '\0'; i++){
    char c = infix[i];

    if(isalnum(c)){ // Operand
       prefix[j++] = c;
    } else if(c == ')') {  // Closing parenthesis
       push(&stack, c);
    } else if(c == '(') { // Opening parenthesis
      while(!isEmpty(&stack) && peek(&stack) != ')'){
        prefix[j++] = pop(&stack);
      }
      if(!isEmpty(&stack) && peek(&stack) == ')') {
        pop(&stack);
      }
    } else if(isOperator(c)) { // Operator
       while(!isEmpty(&stack)   &&   precedence(peek(&stack))   >=
precedence(c)){
         prefix[j++] = pop(&stack);
       }
      push(&stack, c);
    }
  }

  // Pop remaining operators
  while(!isEmpty(&stack)){
    prefix[j++] = pop(&stack);
  }


  prefix[j] = '\0';
```

```
  reverse(prefix); // Reverse the result to get the final prefix expression
}

// Main fuction
int main(){
  char infix[MAX], prefix[MAX];

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

  infixToPrefix(infix, prefix);

  printf("Prefix expression: %s\n", prefix);

  return 0;
}
```

3. Konversikan rumus di bawah ini dalam bentuk Postfix cara stack

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Konversi Rumus Kasus (+) ke Postfix:

Inputan : (0 b - sqrt (b ^ 2 - 4 * a * c) + (2 * a))

Token Dibaca	Stack	Output
(	(	
0	(	0
b	(	0 b
-	(-	0 b
sqrt	(- sqrt	0 b
(	(- sqrt (	0 b
b	(- sqrt (	0 b b
^	(- sqrt (^	0 b b
2	(- sqrt (^	0 b b 2
-	(- sqrt (-	0 b b 2 ^
4	(- sqrt (-	0 b b 2 ^ 4
*	(- sqrt (- *	0 b b 2 ^ 4
a	(- sqrt (- *	0 b b 2 ^ 4 a
*	(- sqrt (- *	0 b b 2 ^ 4 a *
c	(- sqrt (- *	0 b b 2 ^ 4 a * c
)	(-	0 b b 2 ^ 4 a * c * - sqrt
+	(+	0 b b 2 ^ 4 a * c * - sqrt -
(	(+ (	0 b b 2 ^ 4 a * c * - sqrt -
2	(+ (	0 b b 2 ^ 4 a * c * - sqrt - 2
*	(+ (*	0 b b 2 ^ 4 a * c * - sqrt - 2
a	(+ (*	0 b b 2 ^ 4 a * c * - sqrt - 2 a
)	(+	0 b b 2 ^ 4 a * c * - sqrt - 2 a *
/	(/	0 b b 2 ^ 4 a * c * - sqrt - 2 a * +
)	kosong	0 b b 2 ^ 4 a * c * - sqrt - 2 a * + /

Output : 0 b - b 2 ^ 4 a * c * - sqrt + 2 a * /

Konversi Rumus Kasus (-) ke Postfix:

Inputan : (0 b - sqrt (b ^ 2 - 4 * a * c) - (2 * a))

Token Dibaca	Stack	Output
(	(	
0	(	0
b	(	0 b
-	(-	0 b
sqrt	(- sqrt	0 b
(	(- sqrt (	0 b
b	(- sqrt (	0 b b
^	(- sqrt (^	0 b b
2	(- sqrt (^	0 b b 2
-	(- sqrt (-	0 b b 2 ^
4	(- sqrt (-	0 b b 2 ^ 4
*	(- sqrt (- *	0 b b 2 ^ 4
a	(- sqrt (- *	0 b b 2 ^ 4 a
*	(- sqrt (- *	0 b b 2 ^ 4 a *
c	(- sqrt (- *	0 b b 2 ^ 4 a * c
)	(-	0 b b 2 ^ 4 a * c * - sqrt
-	(-	0 b b 2 ^ 4 a * c * - sqrt -
(	(- (	0 b b 2 ^ 4 a * c * - sqrt -
2	(- (	0 b b 2 ^ 4 a * c * - sqrt - 2
*	(- (*	0 b b 2 ^ 4 a * c * - sqrt - 2
a	(- (*	0 b b 2 ^ 4 a * c * - sqrt - 2 a
)	(-	0 b b 2 ^ 4 a * c * - sqrt - 2 a *
/	(/	0 b b 2 ^ 4 a * c * - sqrt - 2 a * -
)	kosong	0 b b 2 ^ 4 a * c * - sqrt - 2 a * - /

Output : 0 b - b 2 ^ 4 a * c * - sqrt - 2 a * /