GROUP MEMBERS:

1. MWANGI ANN MWENDE SCT222-0302/2022
2. BETH WAMUGWE KAMAU SCT222-0297/2022
3. BRENDA MAINA SCT222-0194/2021

Write a program that reads an expression from the standard output device. The device should convert the expression into reversed polish notation (post fix form). The program should then output the RPN expression on standard output device. The program should be able to handle the following operations (+, -, \*, /,%)

SOLUTION

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SIZE 100

typedef struct {

char data[MAX\_SIZE];

int top;

} Stack;

void push(Stack \*stack, char c) {

if (stack->top == MAX\_SIZE - 1) {

printf("Stack overflow\n");

exit(EXIT\_FAILURE);

}

stack->data[++stack->top] = c;

}

char pop(Stack \*stack) {

if (stack->top == -1) {

printf("Stack underflow\n");

exit(EXIT\_FAILURE);

}

return stack->data[stack->top--];

}

int is\_operator(char c) {

return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '%');

}

int precedence(char c) {

if (c == '+' || c == '-') {

return 1;

} else if (c == '\*' || c == '/' || c == '%') {

return 2;

} else {

return 0;

}

}

char \*infix\_to\_rpn(char \*expression) {

Stack stack;

stack.top = -1;

char \*rpn = (char \*)malloc(MAX\_SIZE \* sizeof(char));

int rpn\_index = 0;

for (int i = 0; i < strlen(expression); i++) {

if (expression[i] >= '0' && expression[i] <= '9') {

rpn[rpn\_index++] = expression[i];

} else if (expression[i] == '(') {

push(&stack, expression[i]);

} else if (expression[i] == ')') {

while (stack.top != -1 && stack.data[stack.top] != '(') {

rpn[rpn\_index++] = pop(&stack);

}

if (stack.top == -1) {

printf("Invalid expression\n");

exit(EXIT\_FAILURE);

}

pop(&stack); // Remove '(' from stack

} else if (is\_operator(expression[i])) {

while (stack.top != -1 && precedence(stack.data[stack.top]) >= precedence(expression[i])) {

rpn[rpn\_index++] = pop(&stack);

}

push(&stack, expression[i]);

}

}

while (stack.top != -1) {

if (stack.data[stack.top] == '(') {

printf("Invalid expression\n");

exit(EXIT\_FAILURE);

}

rpn[rpn\_index++] = pop(&stack);

}

rpn[rpn\_index] = '\0';

return rpn;

}

int main() {

char expression[MAX\_SIZE];

printf("Enter the infix expression: ");

fgets(expression, MAX\_SIZE, stdin);

char \*rpn\_expression = infix\_to\_rpn(expression);

printf("Reverse Polish Notation (RPN) expression: %s\n", rpn\_expression);

free(rpn\_expression);

return 0;

}