LAB TASK – 4

1. Implementation of stack using array::-

ANS::-

#include <stdio.h>

#define MAX\_SIZE 100

typedef struct {

int arr[MAX\_SIZE];

int top;

} Stack;

void initStack(Stack \*s) {

s->top = -1;

}

void push(Stack \*s, int data) {

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

printf("Stack Overflow\n");

return;

}

s->arr[++s->top] = data;

}

int pop(Stack \*s) {

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

printf("Stack Underflow\n");

return -1;

}

return s->arr[s->top--];

}

int main() {

Stack s;

initStack(&s);

push(&s, 28);

push(&s, 3);

push(&s, 2006);

printf("Popped element: %d\n", pop(&s));

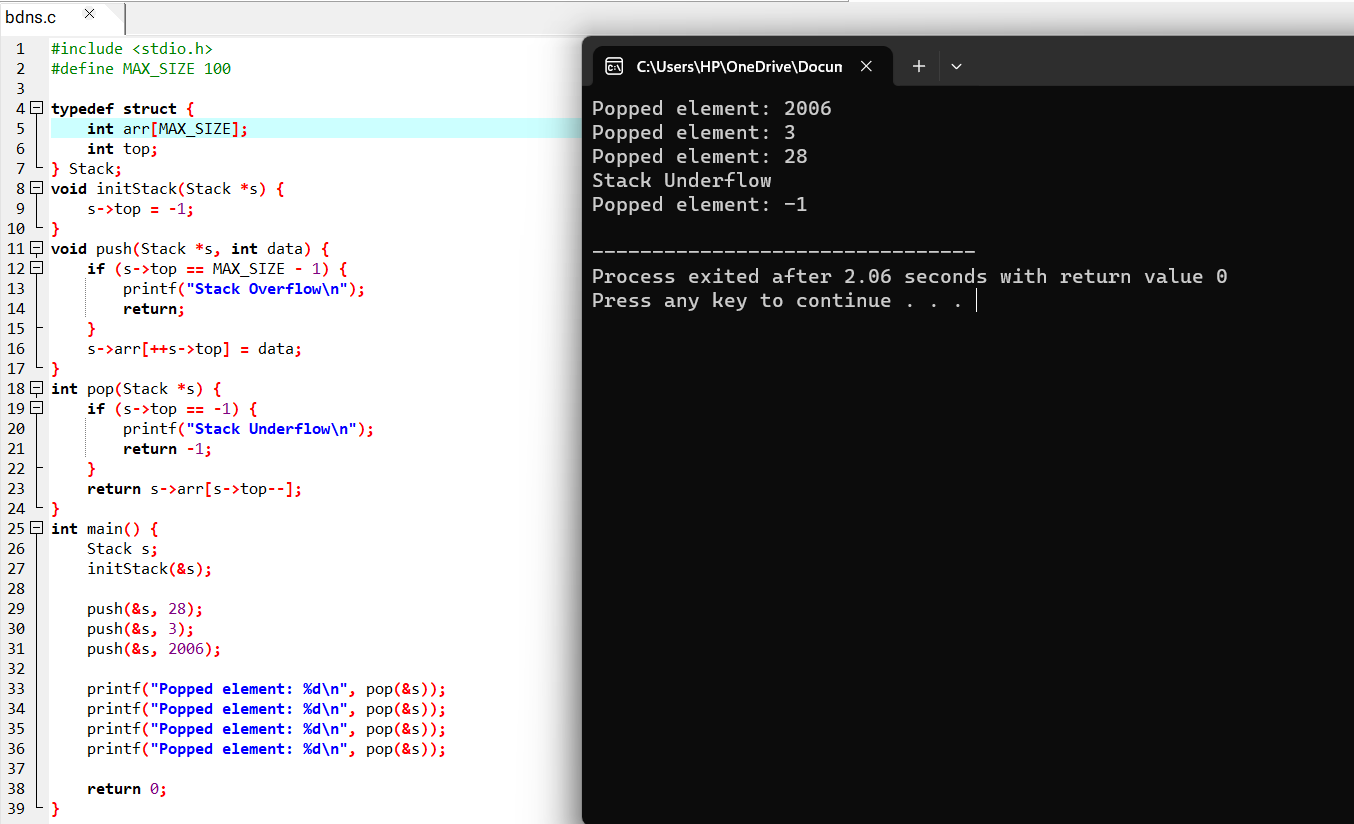
printf("Popped element: %d\n", pop(&s));

printf("Popped element: %d\n", pop(&s));

printf("Popped element: %d\n", pop(&s));

return 0;

}



1. Conversion of infix expression to postfix expression::

ANS::

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_SIZE 100

typedef struct {

char arr[MAX\_SIZE];

int top;

} Stack;

void initStack(Stack \*s) {

s->top = -1;

}

void push(Stack \*s, char data) {

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

printf("Stack Overflow\n");

return;

}

s->arr[++s->top] = data;

}

char pop(Stack \*s) {

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

printf("Stack Underflow\n");

return -1;

}

return s->arr[s->top--];

}

int isOperator(char c) {

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

}

int precedence(char c) {

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

return 2;

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

return 1;

} else {

return 0;

}

}

void infixToPostfix(char \*infix, char \*postfix) {

Stack s;

initStack(&s);

int i = 0, j = 0;

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

if (isalnum(infix[i])) {

postfix[j++] = infix[i];

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

push(&s, infix[i]);

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

while (s.top != -1 && s.arr[s.top] != '(') {

postfix[j++] = pop(&s);

}

if (s.top != -1 && s.arr[s.top] == '(') {

pop(&s);

}

} else {

while (s.top != -1 && precedence(infix[i]) <= precedence(s.arr[s.top])) {

postfix[j++] = pop(&s);

}

push(&s, infix[i]);

}

i++;

}

while (s.top != -1) {

postfix[j++] = pop(&s);

}

postfix[j] = '\0';

}

int main() {

char infix[MAX\_SIZE], postfix[MAX\_SIZE];

printf("Enter infix expression: ");

fgets(infix, MAX\_SIZE, stdin);

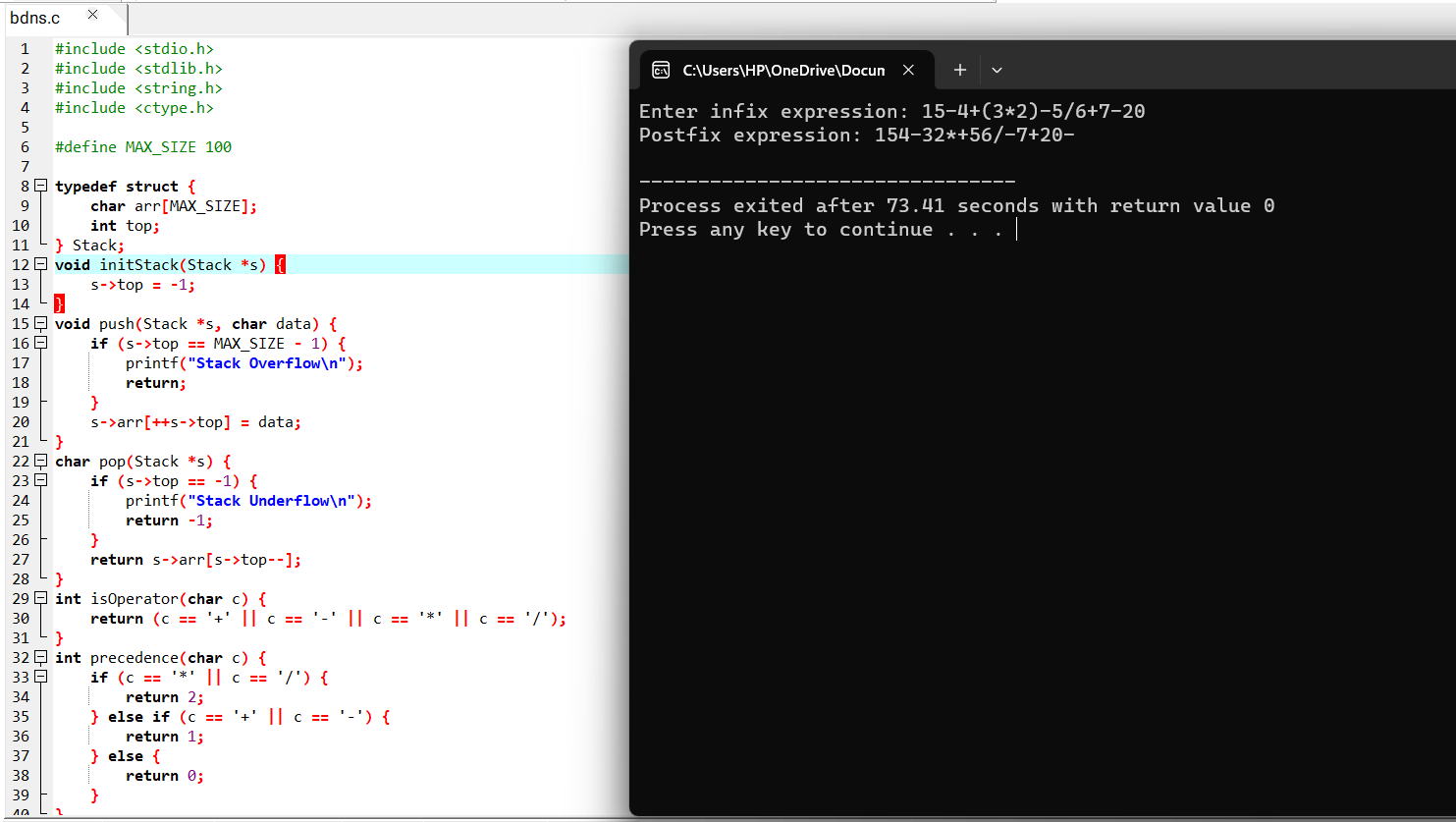
infix[strlen(infix) - 1] = '\0';

infixToPostfix(infix, postfix);

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

return 0;

}



1. Evaluation of expressions::

ANS::

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_SIZE 100

typedef struct {

int arr[MAX\_SIZE];

int top;

} Stack;

void initStack(Stack \*s) {

s->top = -1;

}

void push(Stack \*s, int data) {

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

printf("Stack Overflow\n");

return;

}

s->arr[++s->top] = data;

}

int pop(Stack \*s) {

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

printf("Stack Underflow\n");

return -1;

}

return s->arr[s->top--];

}

int isOperator(char c) {

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

}

int evaluatePostfix(char \*postfix) {

Stack s;

initStack(&s);

int i = 0;

while (postfix[i] != '\0') {

if (isdigit(postfix[i])) {

push(&s, postfix[i] - '0');

} else if (isOperator(postfix[i])) {

int operand2 = pop(&s);

int operand1 = pop(&s);

switch (postfix[i]) {

case '+':

push(&s, operand1 + operand2);

break;

case '-':

push(&s, operand1 - operand2);

break;

case '\*':

push(&s, operand1 \* operand2);

break;

case '/':

push(&s, operand1 / operand2);

break;

}

}

i++;

}

return pop(&s);

}

int main() {

char postfix[MAX\_SIZE];

printf("Enter postfix expression: ");

fgets(postfix, MAX\_SIZE, stdin);

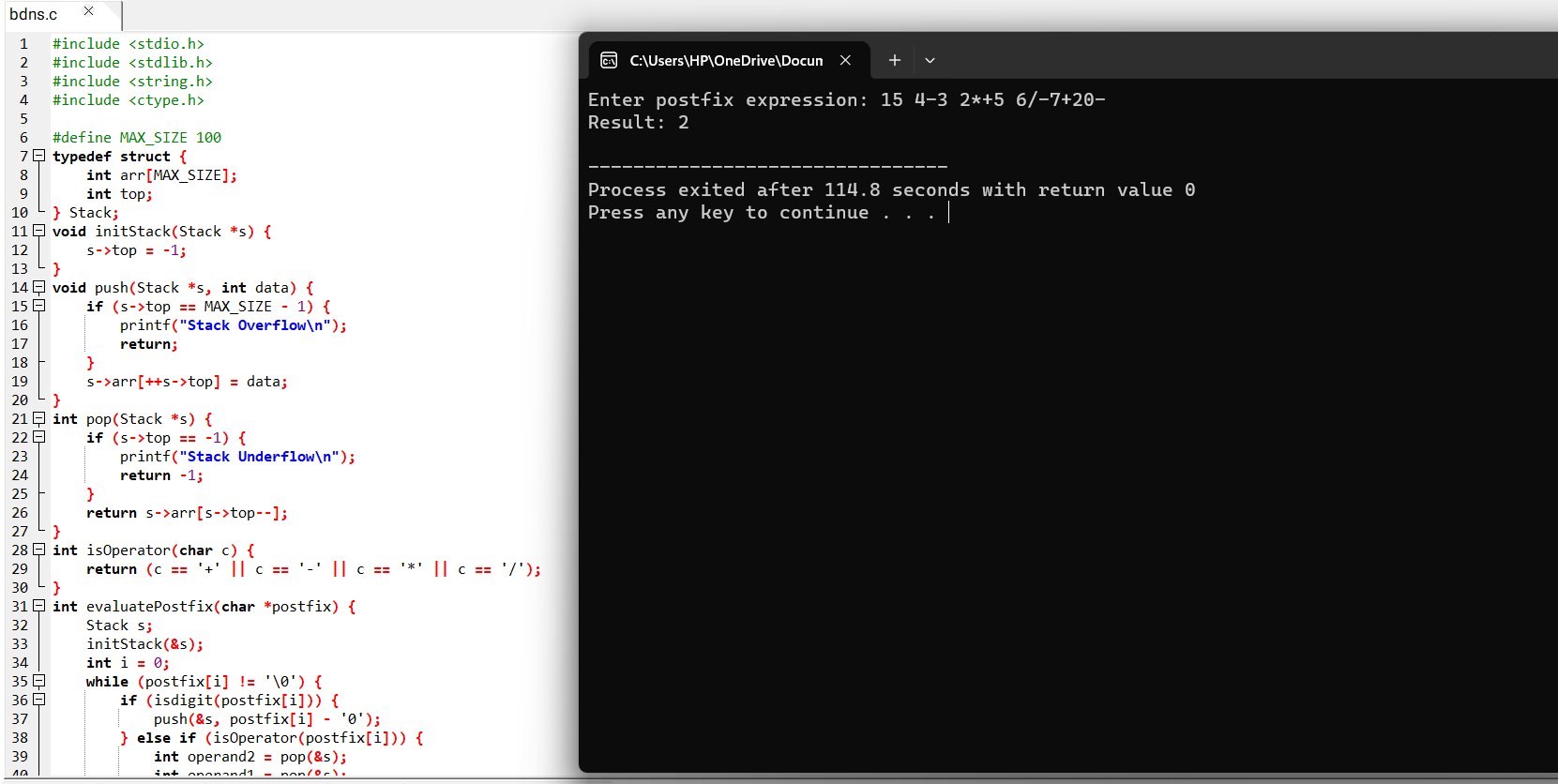
postfix[strlen(postfix) - 1] = '\0';

int result = evaluatePostfix(postfix);

printf("Result: %d\n", result);

return 0;

}



1. Tower of Hanoi::

ANS::

#include <stdio.h>

void move(int n, char source, char destination, char intermediate) {

if (n == 1) {

printf("Move disk 1 from rod %c to rod %c\n", source, destination);

return;

}

move(n - 1, source, intermediate, destination);

printf("Move disk %d from rod %c to rod %c\n", n, source, destination);

move(n - 1, intermediate, destination, source);

}

int main() {

int n;

printf("Enter the number of disks: ");

scanf("%d", &n);

move(n, 'A', 'C', 'B');

return 0;

}