         LAB TASK  - 4

Program – 2:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int top = -1,n=100;

char stack[100];

void push(char c) {

    if (top == n- 1) {

        printf("Stack overflow\n");

    }

    stack[++top] = c;

}

char pop() {

    if (top == -1) {

        printf("Stack underflow\n");

    }

    return stack[top--];

}

int prec(char c) {

    if (c == '+' || c == '-')

        return 1;

    else if (c == '\*' || c == '/')

        return 2;

    else

        return 0;

}

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

    int i = 0, j = 0;

    while (infix[i] != '\0') {

        if (infix[i] >= 'a' && infix[i] <= 'z') {

            postfix[j++] = infix[i];

        } else if (infix[i] == '(') {

            push(infix[i]);

        } else if (infix[i] == ')') {

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

                postfix[j++] = pop();

            }

            pop();

        } else {

            while (top != -1 && prec(stack[top]) >= prec(infix[i])) {

                postfix[j++] = pop();

            }

            push(infix[i]);

        }

        i++;

    }

    while (top != -1) {

        postfix[j++] = pop();

    }

    postfix[j] = '\0';

}

   int main() {

    char infix[n], postfix[n];

    printf("Enter an infix expression: ");

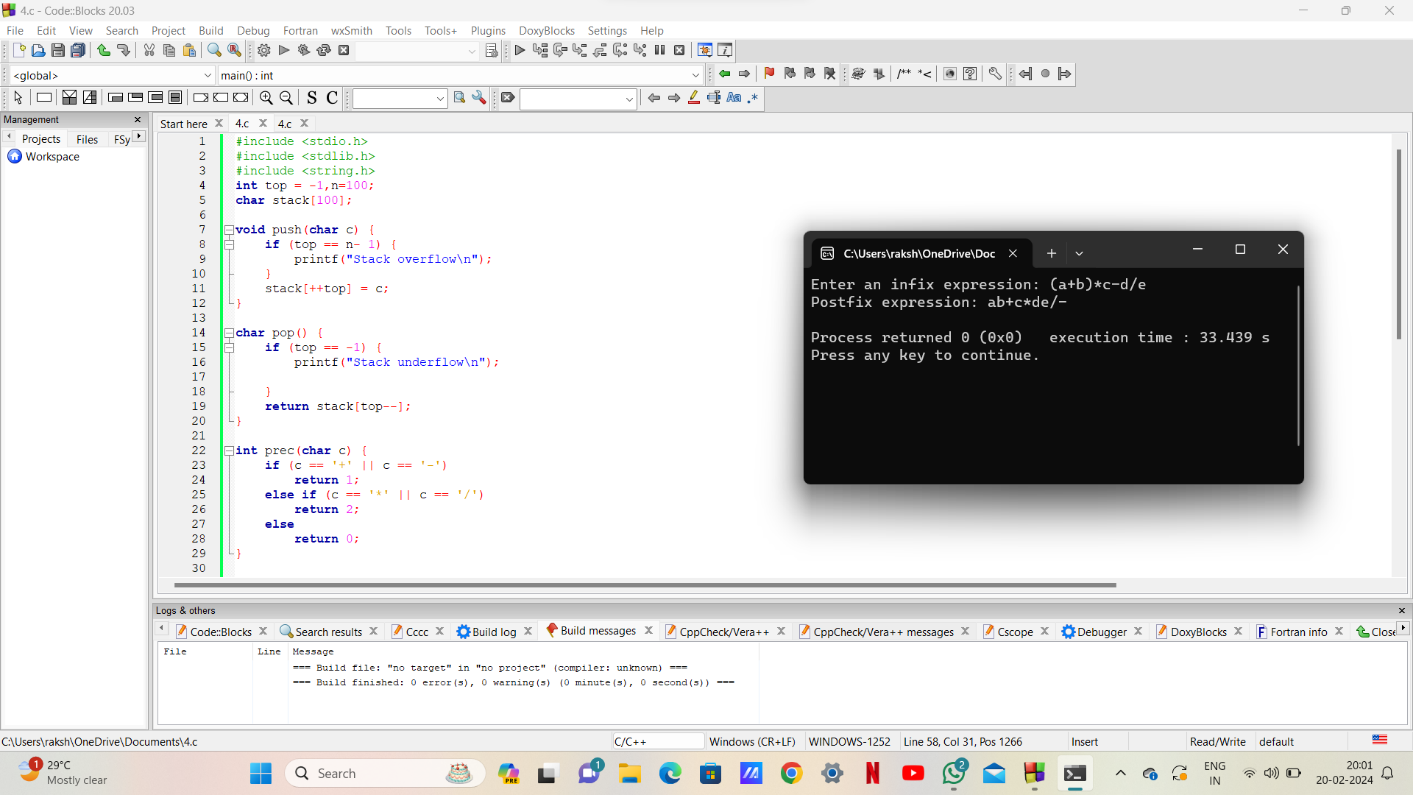
    scanf("%s", infix);

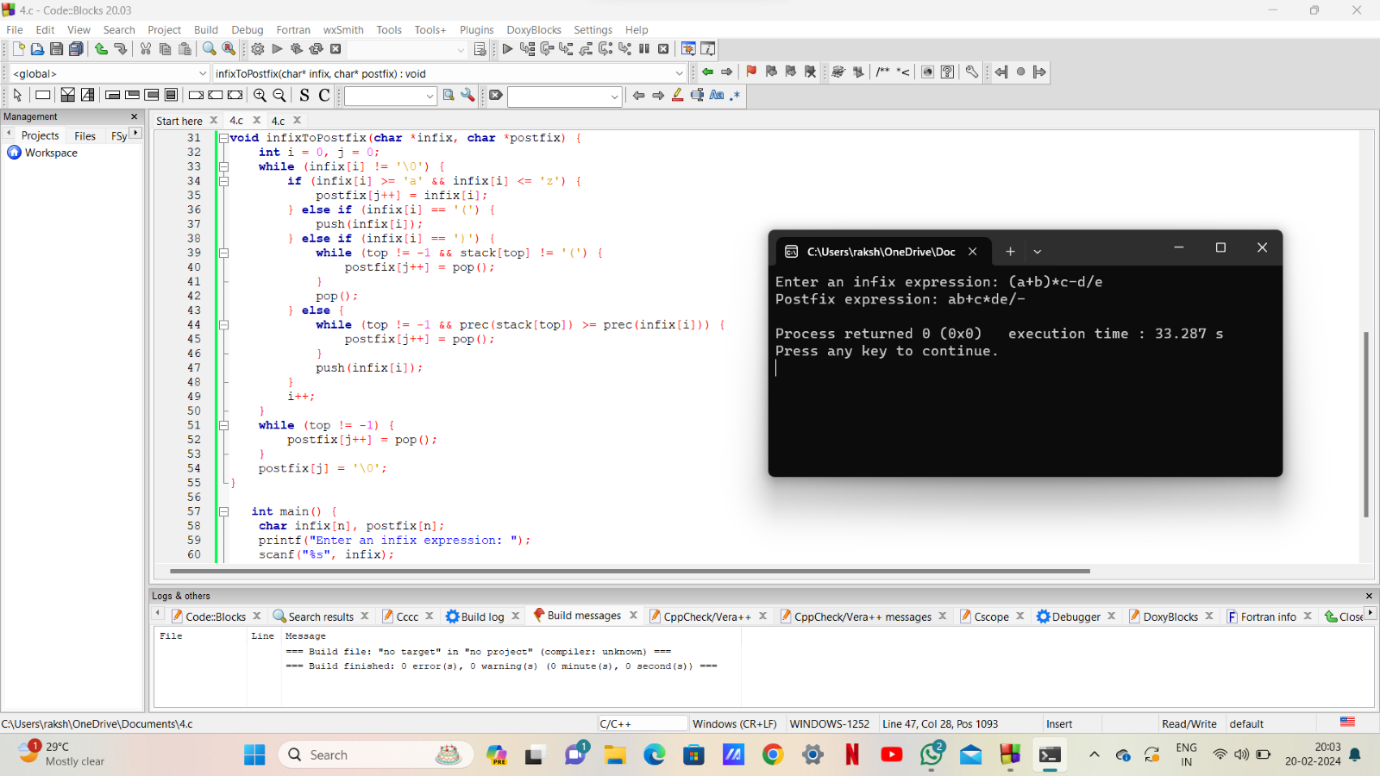
    infixToPostfix(infix, postfix);

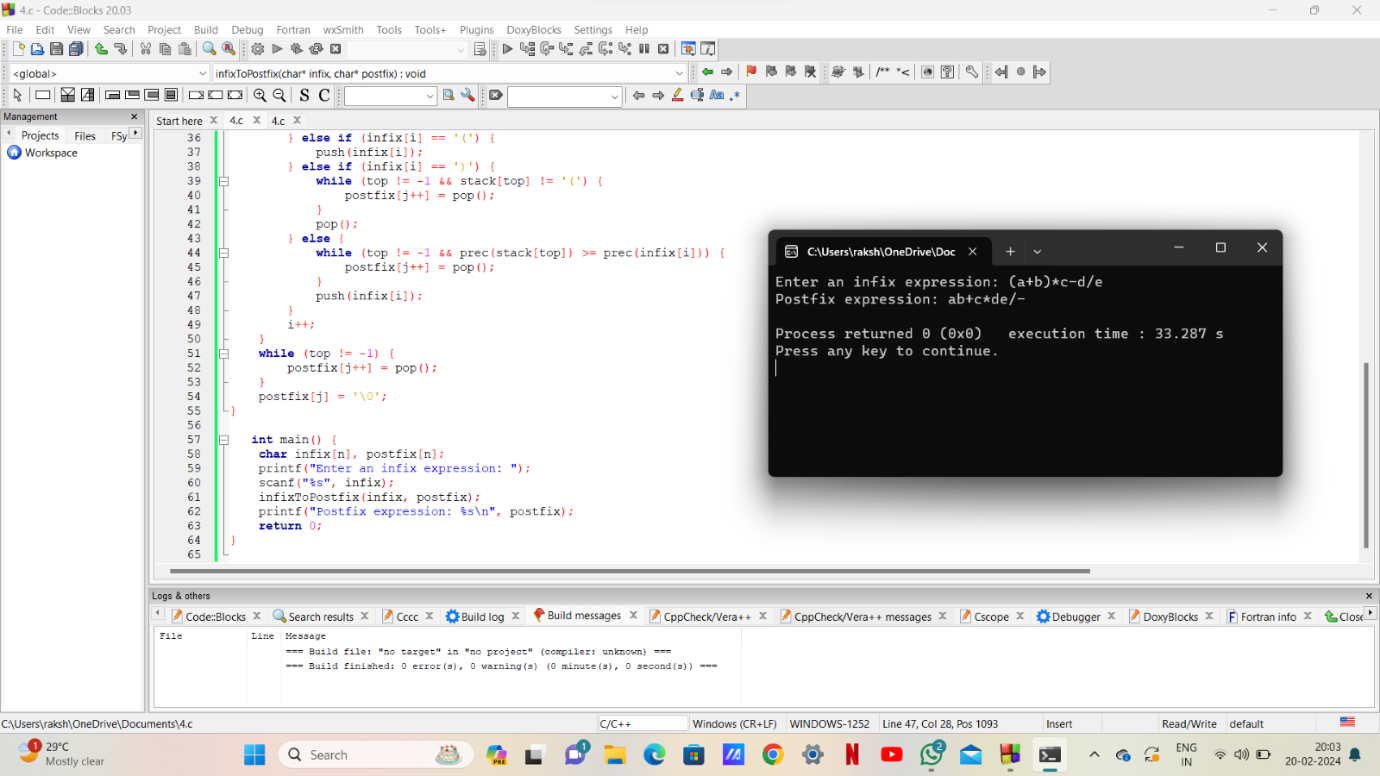
    printf("Postfix expression: %s\n", postfix);

    return 0;

}







Program – 1:

#include<stdio.h>

int n=5,i,stack\_arr[5],top=-1;

 int isfull()

  {

    if(top==n-1)

        return 1;

    else

        return 0;

  }

  int isempty()

  {

   if(top==-1)

        return 1;

   else

    return 0;

  }

void push(int d)

{

    if(isfull())

    {

        printf("stack overflow");

    }

    else{

    top=top+1;

    stack\_arr[top]=d;

    }

}

void print()

{

    if(isempty())

    {

        printf("\nstack underflow");

    }

    printf("\nstack :");

    printf("{");

    for(i=top;i>=0;i--)

    {

        printf("%d,",stack\_arr[i]);

    }

    printf("}");

    printf("\n");

}

void pop()

{

     if(isempty())

    {

        printf("\nstack underflow");

    }

    i=stack\_arr[top];

    top=top-1;

    printf("popped item:%d",i);

}

void peek()

{

    if(isempty())

    {

        printf("underflow");

    }

    else

    {

        printf("Top of stack is:%d\n",stack\_arr[top]);

    }

}

int main()

{

    push(1);

    push(2);

    push(3);

    push(4);

    push(5);

     print();

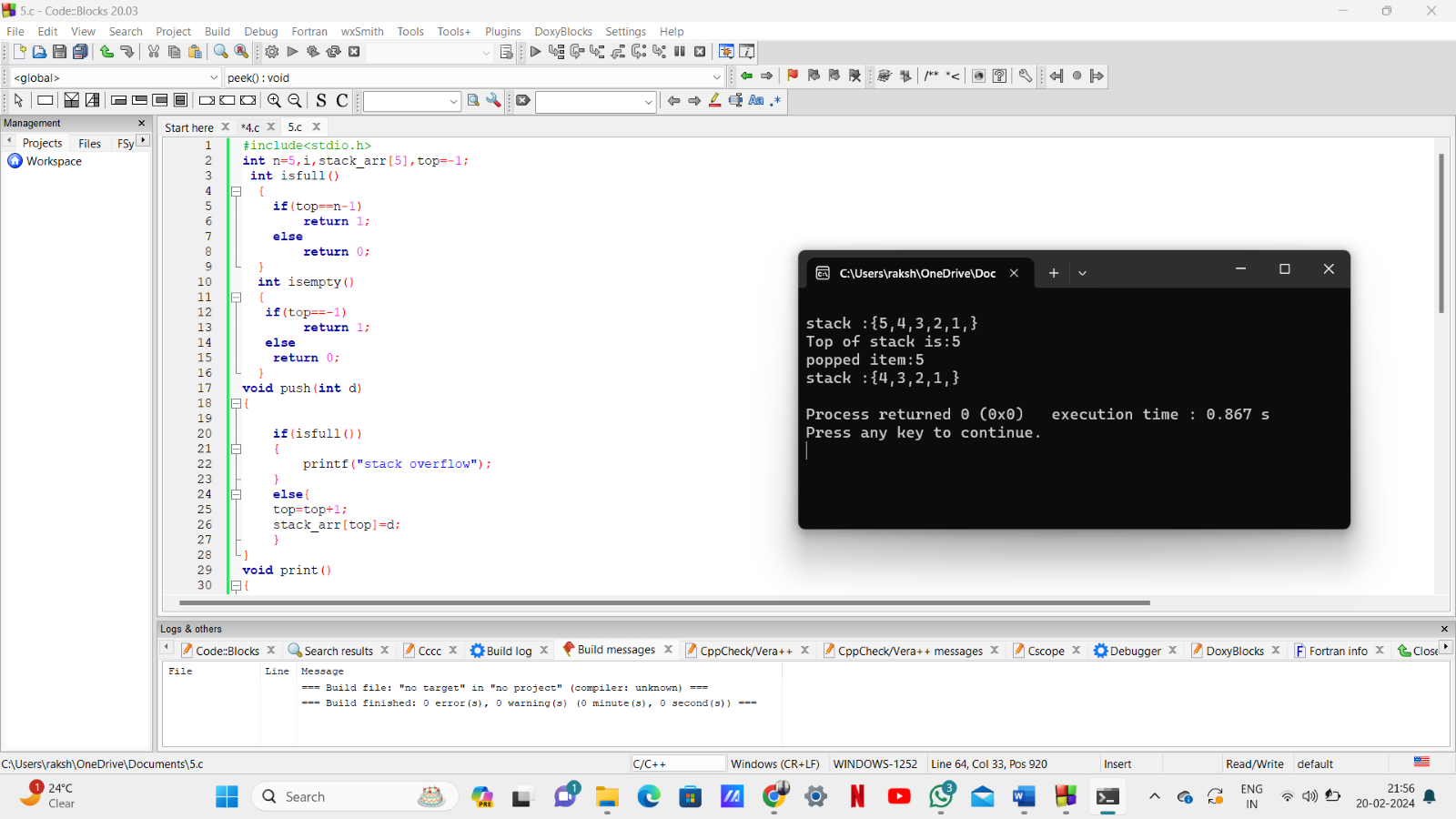
    peek();

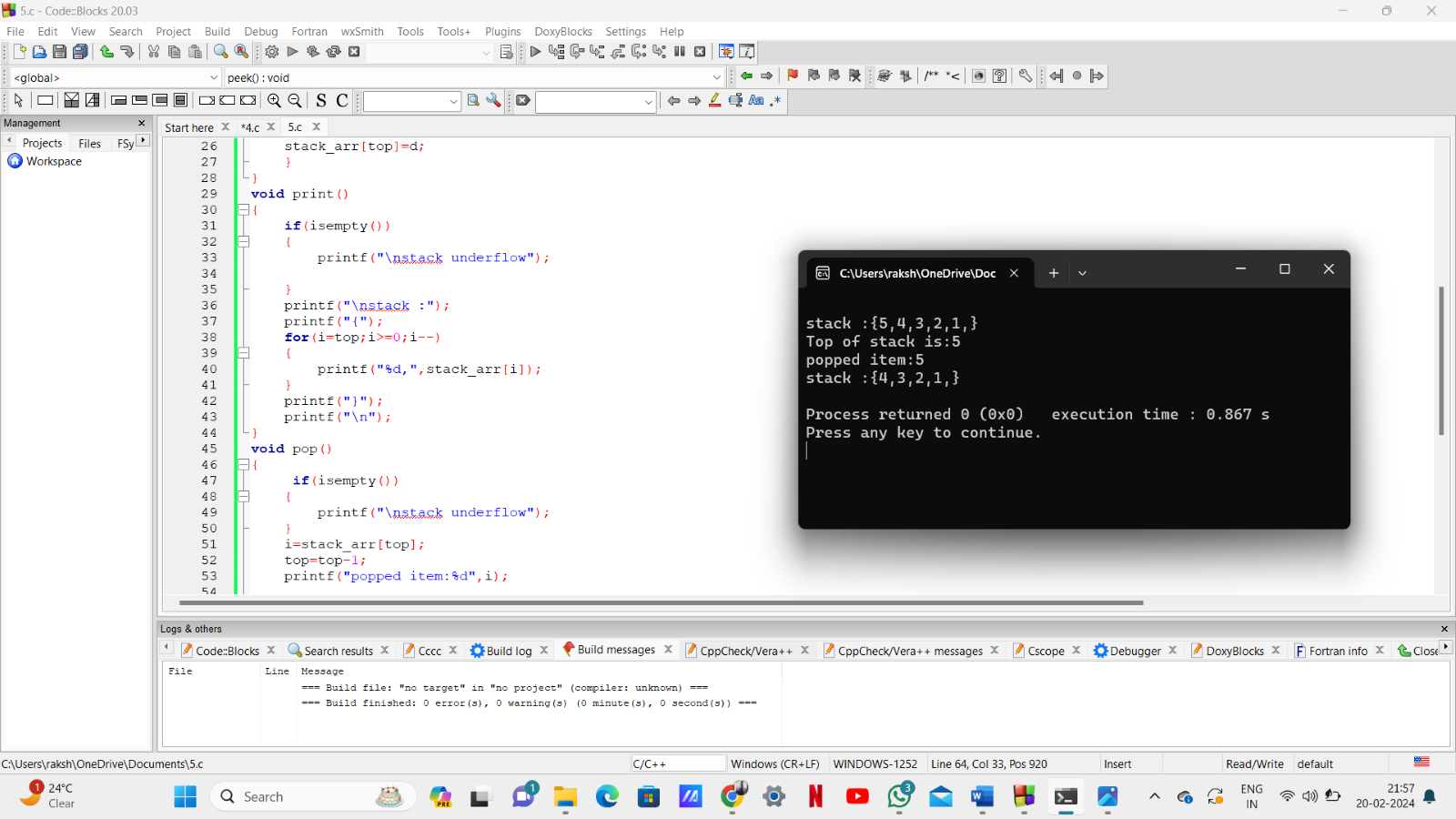
    pop();

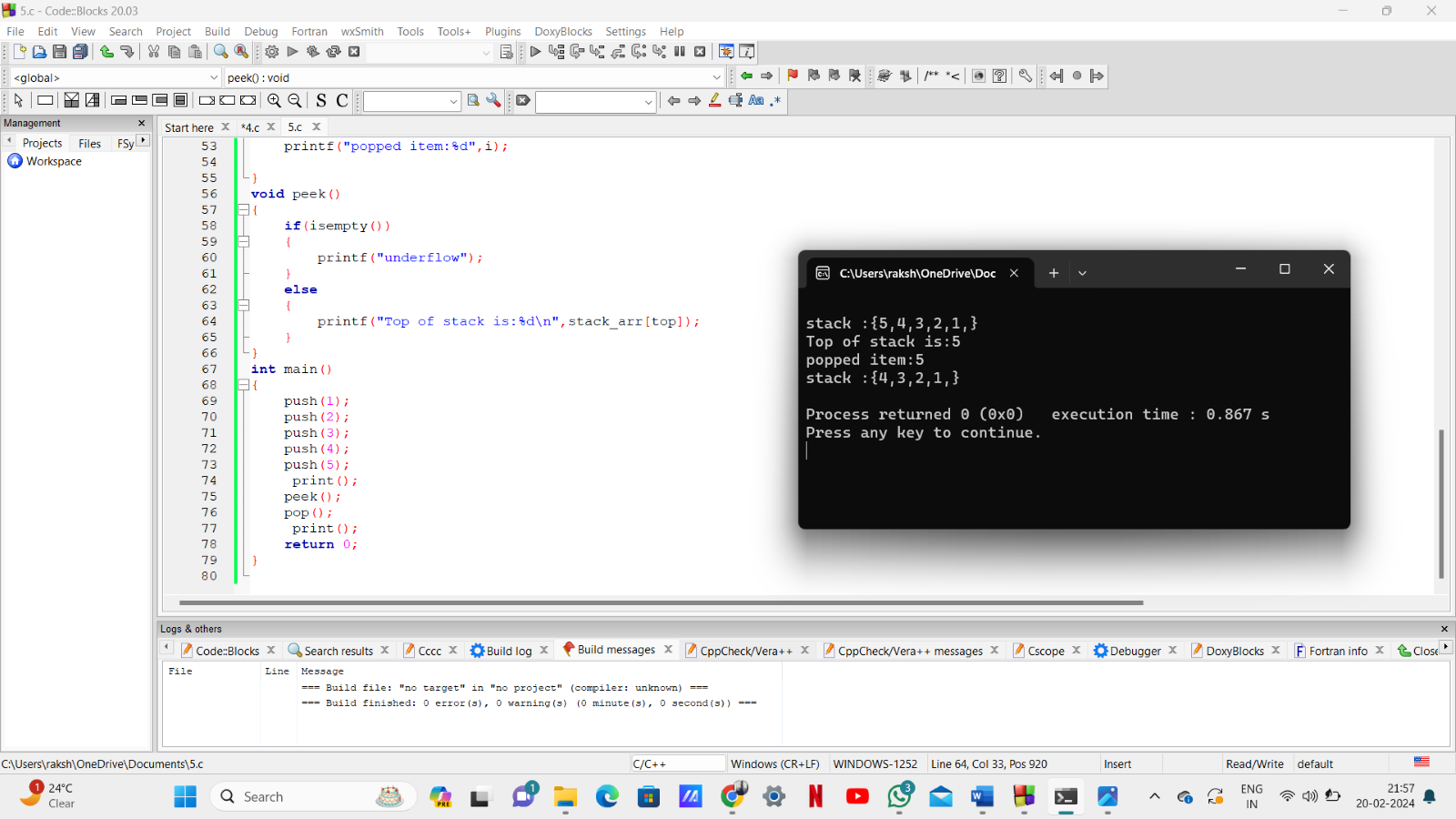
     print();

    return 0;

}







Program –4:

#include <stdio.h>

void move(int n, char strod , char lastrod , char midrod ) {

    if (n == 1) {

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

        return;

    }

    move(n - 1, strod,midrod,lastrod);

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

    move(n - 1, midrod,lastrod,strod);

}

int main() {

    int n;

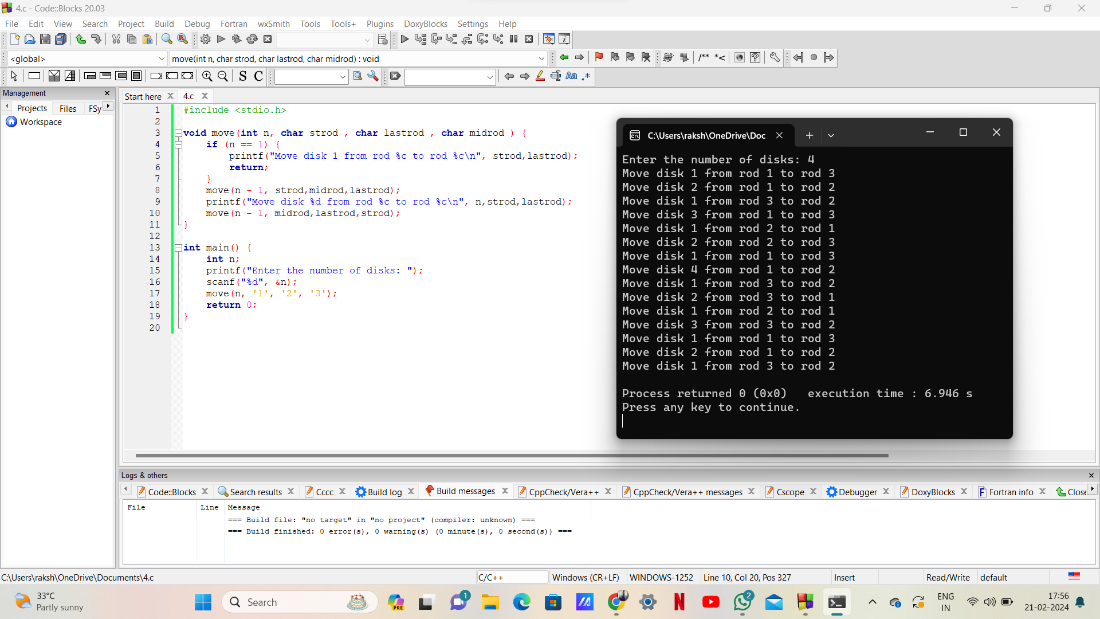
    printf("Enter the number of disks: ");

    scanf("%d", &n);

    move(n, '1', '2', '3');

    return 0;

}



Program – 3:

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

int n=100;

int createStack() {

    return -1;

}

int isFull(int top) {

    return top == n - 1;

}

int isEmpty(int top) {

    return top == -1;

}

void push(int stack[], int\* top, char item) {

    if (isFull(\*top))

        return;

    stack[++(\*top)] = item;

}

char pop(int stack[], int\* top) {

    if (isEmpty(\*top))

        return '\0';

    return stack[(\*top)--];

}

char peek(int stack[], int top) {

    if (isEmpty(top))

        return '\0';

    return stack[top];

}

int isOperator(char ch) {

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

}

int precedence(char ch) {

    if (ch == '+' || ch == '-')

        return 1;

    else if (ch == '\*' || ch == '/')

        return 2;

    else

        return 0;

}

void infixToPostfix(char infix[], char postfix[]) {

    int stack[n];

    int top = createStack();

    int i = 0, j = 0;

    while (infix[i] != '\0') {

        if (isalnum(infix[i]))

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

        else if (infix[i] == '(')

            push(stack, &top, infix[i++]);

        else if (infix[i] == ')') {

            while (!isEmpty(top) && peek(stack, top) != '(')

                postfix[j++] = pop(stack, &top);

            pop(stack, &top);

            i++;

        } else {

            while (!isEmpty(top) && precedence(infix[i]) <= precedence(peek(stack, top)))

                postfix[j++] = pop(stack, &top);

            push(stack, &top, infix[i++]);

        }

    }

    while (!isEmpty(top))

        postfix[j++] = pop(stack, &top);

    postfix[j] = '\0';

}

int evaluatePostfix(char postfix[]) {

    int stack[n];

    int top = createStack();

    int i = 0;

    while (postfix[i] != '\0') {

        if (isdigit(postfix[i]))

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

        else {

            int operand2 = pop(stack, &top);

            int operand1 = pop(stack, &top);

            switch (postfix[i]) {

                case '+': push(stack, &top, operand1 + operand2); break                case '-': push(stack, &top, operand1 - operand2); break;

                case '\*': push(stack, &top, operand1 \* operand2); break;

                case '/': push(stack, &top, operand1 / operand2); break;

            }

        }

        i++;

    }

    return pop(stack, &top);

}

int main() {

    char infix[n];

    printf("Enter infix expression: ");

    scanf("%s", infix);

    char postfix[n];

    infixToPostfix(infix, postfix);

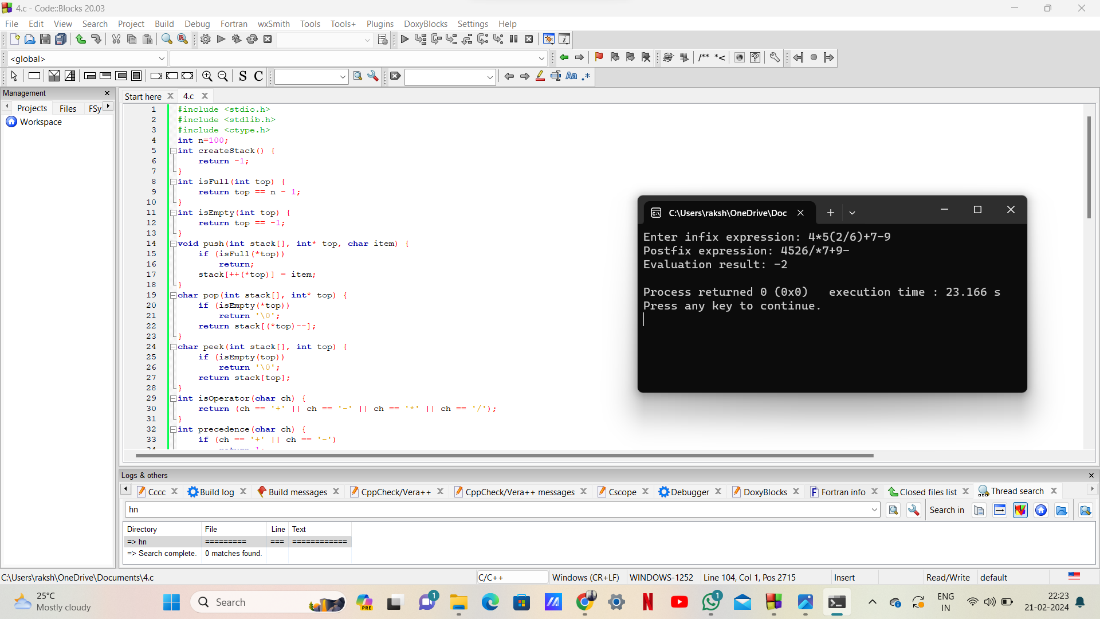
    printf("Postfix expression: %s\n", postfix);

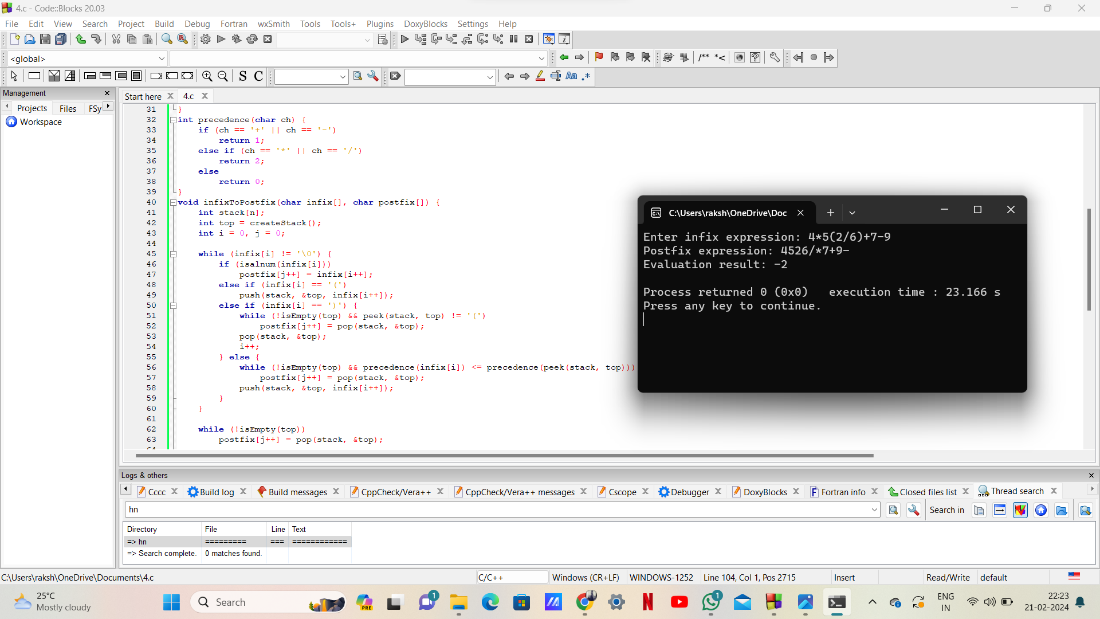
    int result = evaluatePostfix(postfix);

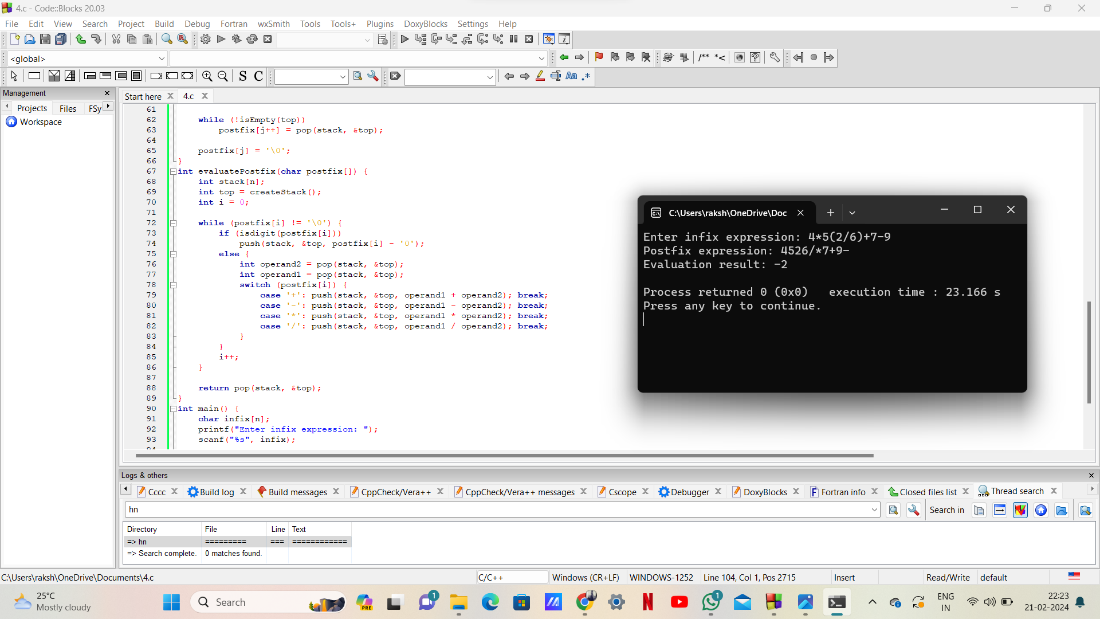
    printf("Evaluation result: %d\n", result);

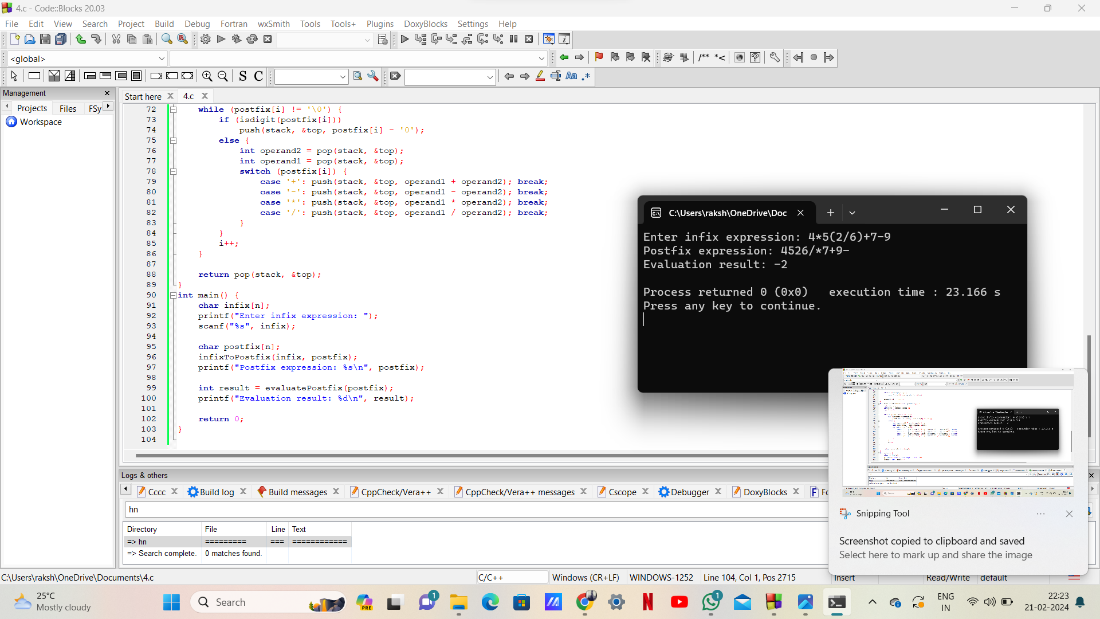
    return 0;

}









   \*\*\*\*THE END\*\*\*\*