**Lab 5 codes:**

**Task 1:**

void reverse\_num\_array(int \* nu\_array, int arr\_size)

{

struct node \* top = NULL;

struct element d4;

struct element d5;

d4.d= num\_array[arr\_size];

for (int i=0; i<arr-size; i++)

{

d4.d= num\_array[i];

push(&top,d4);

}

for (int i=0; i<arr\_size; i++)

{

d5=pop(&top);

num\_array[i]= d5.d;

}

}

**Task 2:**

int isBalanced(char \* ptr\_array)

{

struct element d7;

for(int i=0;ptr\_array!=\0;i+=)

{

if(ptr\_array[i]=='(' || ptr\_array[i]='[' || ptr\_array[i]=='{')

push(&d7,exp[i]);

else if(ptr\_array[i]==')' || ptr\_array[i]==']' || ptr\_array[i]=='}')

pop(d7);

if (d7 == NULL)

return 0;

else if (!isMatchingPair(pop(&d7), ptr\_array[i]))

return 0;

}

}

**Task 3:**

**bool isMatchingPair(char character1, char character2)**

**{**

**if (character1 == '(' && character2 == ')')**

**return 1;**

**else if (character1 == '{' && character2 == '}')**

**return 1;**

**else if (character1 == '[' && character2 == ']')**

**return 1;**

**else**

**return 0;**

**}**

**void infixToPostfix(char \*src, char \*dst)**

**{**

**if (!isBalanced(src)) {**

**printf("Unbalanced parentheses in the expression.\n");**

**return;**

**}**

**int src\_len = strlen(src);**

**char stack[MAX\_EXPRESSION\_SIZE];**

**int stack\_top = -1;**

**int j = 0;**

**for (int i = 0; i < src\_len; i++) {**

**char currentChar = src[i];**

**if (currentChar >= '0' && currentChar <= '9') {**

**dst[j++] = currentChar;**

**} else if (isOperator(currentChar)) {**

**while (stack\_top >= 0 && isOperator(stack[stack\_top]) &&**

**precedence(stack[stack\_top]) >= precedence(currentChar)) {**

**dst[j++] = stack[stack\_top--];**

**}**

**stack[++stack\_top] = currentChar;**

**} else if (currentChar == '(') {**

**stack[++stack\_top] = currentChar;**

**} else if (currentChar == ')') {**

**while (stack\_top >= 0 && stack[stack\_top] != '(') {**

**dst[j++] = stack[stack\_top--];**

**}**

**if (stack\_top >= 0 && stack[stack\_top] == '(') {**

**stack\_top--; // Pop the opening parenthesis**

**} else {**

**printf("Unbalanced parentheses in the expression.\n");**

**return;**

**}**

**}**

**}**

**while (stack\_top >= 0) {**

**if (stack[stack\_top] == '(') {**

**printf("Unbalanced parentheses in the expression.\n");**

**return;**

**}**

**dst[j++] = stack[stack\_top--];**

**}**

**dst[j] = '\0';**

**}**

**int isOperator(char c) {**

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

**}**

**int precedence(char operator) {**

**if (operator == '+' || operator == '-') {**

**return 1;**

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

**return 2;**

**}**

**return 0;**

**}**