TOPIC NO.: 8

TITLE: Postfix To Infix

Program Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_SIZE 100

typedef struct StackNode

{

char \*data;

struct StackNode \*next;

} StackNode;

StackNode \*createNode(char \*data)

{

StackNode \*newNode = (StackNode \*)malloc(sizeof(StackNode));

if (newNode == NULL)

{

printf("Memory Allocation Failed!\n");

exit(EXIT\_FAILURE);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

int isOperator(char ch)

{

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

}

char \*postfixToInfix(char postfix[])

{

StackNode \*stack = NULL;

int i;

for (i = 0; postfix[i] != '\0'; i++)

{

if (isalnum(postfix[i]))

{

char \*operand = (char \*)malloc(2 \* sizeof(char));

Output:

Enter The Postfix Expression: abcd/ef\*-h+\*+

Infix Expression: (a+(b\*(((c/d)-(e\*f))+h)))

DATE: 06.03.24

operand[0] = postfix[i];

operand[1] = '\0';

StackNode \*node = createNode(operand);

node->next = stack;

stack = node;

}

else if (isOperator(postfix[i]))

{

char \*operand2 = stack->data;

stack = stack->next;

char \*operand1 = stack->data;

stack = stack->next;

char \*infix = (char \*)malloc((strlen(operand1) + strlen(operand2) + 4) \* sizeof(char));

sprintf(infix, "(%s%c%s)", operand1, postfix[i], operand2);

StackNode \*node = createNode(infix);

node->next = stack;

stack = node;

}

}

return stack->data;

}

int main()

{

char postfix[MAX\_SIZE];

char \*infix;

printf("Enter The Postfix Expression: ");

fgets(postfix, MAX\_SIZE, stdin);

postfix[strcspn(postfix, "\n")] = '\0';

infix = postfixToInfix(postfix);

printf("Infix Expression: %s\n", infix);

free(infix);

return 0;

}

TOPIC NO.: 9

TITLE: Merge Sort

Program Code:

#include <stdio.h>

void merge(int arr[], int left, int mid, int right)

{

int i, j, k;

int n1 = mid - left + 1;

int n2 = right - mid;

int L[n1], R[n2];

for (i = 0; i < n1; i++)

L[i] = arr[left + i];

for (j = 0; j < n2; j++)

R[j] = arr[mid + 1 + j];

i = 0;

j = 0;

k = left;

while (i < n1 && j < n2)

{

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

}

k++;

}

Output:

Given Array:

12 11 13 5 6 7

Sorted Array:

5 6 7 11 12 13

DATE: 13.03.24

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

}

void mergeSort(int arr[], int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

int arr\_size = sizeof(arr) / sizeof(arr[0]);

printf("Given Array: \n");

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

printf("%d \t", arr[i]);

mergeSort(arr, 0, arr\_size - 1);

printf("\n\nSorted Array: \n");

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

printf("%d \t", arr[i]);

}