Name:Chavda Mamta

Sub:ss(practical)

Roll No:3100

Div:B

Question1

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <conio.h> // TurboC specific header for console I/O

char kw[32][10] = {"int", "float", "while", "for", "do", "char",
"break","auto", "continue", "default", "double", "if", "else","enum", "goto",
"long", "switch", "typedef", "union","unsigned", "void", "volatile", "extern",
"case","const", "return", "sizeof", "static", "struct","register", "signed"};

char op[15] = {'+', '-', '*', '/', '=', ':', ';', '<', '>',','};

char identifiers[20][10]; // Global array to store identifiers
char constants[20][10]; // Global array to store constants
int ic = 0, cc = 0; // Global counters for identifiers and constants

void analyzeString(char str[]);

int main() {
//FILE *file;
char str[100];

FILE *file = fopen("input.txt", "w");

if (file == NULL) {
printf("Error opening the file.\n");

getch(); // Wait for a key press
return 1; // Return an error code
}

while (fgets(str, sizeof(str), file) != NULL) {
analyzeString(str);
}

fclose(file);

getch(); // Wait for a key press before closing the console window
return 0;
}

void analyzeString(char str[]) {
char *ptr;
int i, j;

ptr = strtok(str, " \n");
```

```
while (ptr != NULL) {
int flag = 0;

for (i = 0; i < 32; i++) {
if (strcmp(ptr, kw[i]) == 0) {
printf("KW#%d ", i + 1);
flag = 1;

break;
}
}

if (flag == 0) {
for (j = 0; j < 10; j++) {
if (ptr[0] == op[j]) {
printf("OP#%d ", j + 1);
flag = 1;
break;
}
}
}

if (flag == 0) {
if (isalpha(ptr[0])) {
int isRepeated = 0;
for (i = 0; i < ic; i++) {
if (strcmp(ptr, identifiers[i]) == 0) {
printf("ID#%d ", i + 1);
isRepeated = 1;
break;
}
}
if (!isRepeated) {
strcpy(identifiers[ic++], ptr);
printf("ID#%d ", ic);

}
} else if (isdigit(ptr[0])) {
int isRepeated = 0;
for (i = 0; i < cc; i++) {
if (strcmp(ptr, constants[i]) == 0) {
printf("CO#%d ", i + 1);
isRepeated = 1;
break;
}
}
if (!isRepeated) {
```

```
strcpy(constants[cc++], ptr);
printf("CO#%d ", cc);
}
}
}

ptr = strtok(NULL, " \n");
}
}
```

Question2

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

char stat_table[6][4][10] = {
{"stat", "letter", "digit", "."},
{"start", "id", "int", "error"},
{"id", "id", "id", "error"},
{"int", "error", "int", "s"},
{"s", "error", "real", "error"},
{"real", "error", "real", "error"},
};

void main() {
char input[20], column_stat[10], current_stat[10], next_stat[10];
char ch, choice;
int error, i, c, r, len;

do {
printf("Enter identifier: ");
scanf("%s", input);
len = strlen(input);
strcpy(current_stat, "start");
error = 0; // Reset the error before each input

//for (i = 0; i < len && !error; i++)
for (i = 0; i < len; i++)
{

ch = input[i];

if (isalpha(ch)) {
strcpy(column_stat, "letter");
} else if (isdigit(ch)) {
strcpy(column_stat, "digit");
} else if (ch == '.') {
strcpy(column_stat, ".");
} else {
strcpy(next_stat, "error");
error = 1;
break; // Exit the loop immediately when encountering an error
}

for (r = 1; r < 6; r++) {
```

```
if (strcmp(stat_table[r][0], current_stat) == 0) {
for (c = 1; c < 4; c++) {
if (strcmp(stat_table[0][c], column_stat) == 0) {
strcpy(next_stat, stat_table[r][c]);
break;
}
}

if (strcmp(next_stat, "error") == 0) {
error = 1; // Set error to 1 to break out of the loop
break;
}

printf("%s %c %s\n", current_stat, ch, next_stat);

strcpy(current_stat, next_stat);
break;
}
}
}

if (error) {
printf("\nInvalid Token");
} else {
printf("\nValid");

if (strcmp(current_stat, "id") == 0) {
printf("\nIt is an identifier");
} else if (strcmp(current_stat, "int") == 0) {
printf("\nIt is an integer");
} else if (strcmp(current_stat, "real") == 0) {
printf("\nIt is a real");
}
}

printf("\n\nDo you want to continue? (enter 'y' for yes and 'n' for no): ");
scanf(" %c", &choice);
} while (choice != 'n');



}
```

Question3:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct treenode
{
char info;
struct treenode *left;
struct treenode *right;
}*temp,*a,*b,*c,*d,*temp1,*root;
typedef struct treenode node;
node * proc_e(char input[]);
node * proc_t(char input[]);
node * proc_v(char input[]);
void traversal(node *temp);
int ssm=0;
void main()
{
char input[20];
ssm=0;
clrscr();
printf("Enter String:");
gets(input);
root=proc_e(input);
printf("Parser Tree: ");
traversal(root);
getch();
}

node * proc_e(char input[])
{
char ch;
a=proc_t(input);
while(input[ssm]=='+' || input[ssm]=='-')
{
ch=input[ssm];
ssm++;
b=proc_t(input);
temp=(node *)malloc(sizeof(node));
temp->info=ch;
temp->left=a;
temp->right=b;
a=temp;
}
return a;
}
node * proc_t(char input[])
```

```
{
char ch;
c=proc_v(input);
ssm+=1;
while(input[ssm]=='*' || input[ssm]=='/')
{
ch=input[ssm];
ssm++;
d=proc_v(input);

temp=(node *)malloc(sizeof(node));
temp->info=ch;
temp->left=c;
temp->right=d;
c=temp;
ssm+=1;
}
return c;
}
node * proc_v(char input[])
{
if(isalpha(input[ssm]))
{
temp=(node *)malloc(sizeof(node));
temp->info=input[ssm];
temp->left=NULL;
temp->right=NULL;
return temp;
}
else
{
printf("Error %c",input[ssm]);
exit(0);
}
}
void traversal(node *temp1)
{

if(temp1!=NULL)
{
traversal(temp1->left);
printf("%c",temp1->info);
traversal(temp1->right);
}
}
```