**Task 1:**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

bool isOperator(char opr);

int main()

{

char opr;

printf("Enter any operator: ");

scanf("%c",&opr);

bool result = isOperator(opr);

return 0;

}

bool isOperator(char opr){

switch(opr){

case '\*':

case '-':

case '+':

case '/':

printf("It is an operator");

return true;

default:

printf("It is not an operator");

return false;

}

}

**Output:**

****

**Task 2:**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

/\*

Author: Ali Hassaan Mughal

Task: Check if it is a valid identifier / variable.

Lab2: Compiler Construction Task 2

\*/

bool isIdentifier(char identifier[]);

int main()

{

char identifier[30];

printf("Enter an identifier: ");

scanf("%s", identifier);

bool result = isIdentifier(identifier);

return 0;

}

bool isIdentifier(char identifier[]){

int i;

//First - Ending Character

if (!(identifier[i] == '\_' || (identifier[i] >= 'A' && identifier[i] <= 'Z') || (identifier[i] >= 'a' && identifier[i] <= 'z')))

{

printf("%s: is an invalid identifier",identifier);

return false;

}

//Second - Ending Character

for (i=1;identifier[i]!='\0';i++)

{

if (!(identifier[i] == '\_' || (identifier[i] >= 'A' && identifier[i] <= 'Z') || (identifier[i] >= 'a' && identifier[i] <= 'z') || (identifier[i] >= '0' && identifier[i] <= '9')))

{

printf("%s: is an invalid identifier",identifier);

return false;

}

}

printf("%s: is a valid identifier",identifier);

return true;

}

**Output:**

****

**Task 3:**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

/\*

Enter a string and differentiate the identifiers and operators

\*/

bool isSpecialCharacter(char sp);

bool isIdentifier(char identifier[],int,int);

int main()

{

int i,j,k,idenIndex=0,spIndex = 0;

bool spAlready=false;

char identifierList[5][20] = {"","","","",""};

char spList[100];

char inputlexical[40];

int tmplastvar=0,linescount = 1;

printf("Enter the program: ");

scanf("%[^\n]s", inputlexical);

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

{

if (inputlexical[i] == '\n')

linescount++;

if (isSpecialCharacter(inputlexical[i]) || inputlexical[i] == ' ' || inputlexical[i] == '\t')

{

//printf("starting var: %c %c \n", inputlexical[tmplastvar], inputlexical[i-1]);

if (isIdentifier(inputlexical,tmplastvar,i-1))

{

for (j=tmplastvar;j<=i-1;j++)

{

identifierList[idenIndex][j-tmplastvar] = inputlexical[j];

}

//printf("identifier %s\n", identifierList[idenIndex]);

idenIndex++;

}

tmplastvar = i+1;

//if special character already there don't add it

for (k=0;k<100;k++)

{

if (inputlexical[i] == spList[i])

{

spAlready=true;

}

}

if (!spAlready && !(inputlexical[i] == ' ' || inputlexical[i] == '\t'))

{

spList[spIndex] = inputlexical[i];

spIndex++;

spAlready=false;

}

}

if (inputlexical[i+1] == '\0')

{

if (isIdentifier(inputlexical,tmplastvar,i))

{

for (j=tmplastvar;j<=i;j++)

{

identifierList[idenIndex][j-tmplastvar] = inputlexical[j];

//printf("%d %d", j-tmplastvar, j);

}

//printf("identifier %s\n", identifierList[idenIndex]);

idenIndex++;

}

}

}

printf("The keywords and identifiers are:\n");

for (i=0;i<idenIndex;i++){

if (isSpecialCharacter(inputlexical[0]) && i == 0)

continue;

printf("%s in an identifier\n",identifierList[i]);

}

printf("Special characters are ");

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

printf("%c", spList[i]);

printf("\nThe number of lines: %d", linescount);

return 0;

}

bool isSpecialCharacter(char opr){

if (!(opr == '\_' || (opr >= 'A' && opr <= 'Z') || (opr >= 'a' && opr <= 'z') || (opr >= '0' && opr <= '9')))

return true;

else

return false;

}

bool isIdentifier(char identifier[],int firstvar,int lastvar){

int i;

//First - Ending Character

if (!(identifier[firstvar] == '\_' || (identifier[firstvar] >= 'A' && identifier[firstvar] <= 'Z') || (identifier[firstvar] >= 'a' && identifier[firstvar] <= 'z')))

{

return false;

}

//Second - Ending Character

for (i=firstvar+1;i<=lastvar;i++)

{

if (!(identifier[i] == '\_' || (identifier[i] >= 'A' && identifier[i] <= 'Z') || (identifier[i] >= 'a' && identifier[i] <= 'z') || (identifier[i] >= '0' && identifier[i] <= '9')))

{

return false;

}

}

return true;

}

**Output:**

