COMPILER DESIGN LAB
D1+TD1 SLOT SJT-604

Siddhant Bhagat 22BCE0682

LAB TASK 1

Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#define MAX_TOKEN_SIZE 100

const char* keywords[] = {

   "auto",  "break",  "case",  "char",  "const",  "continue",  "default",  "do",
"double", "else",

   "enum",  "extern",  "float",  "for",  "goto",  "if",  "int",  "long",  "register",
"return",

   "short", "signed", "sizeof", "static", "struct", "switch", "typedef", "union",
"unsigned", "void", "volatile", "while", "scanf"

};

const char* punctuators[] = {

   "{", "}", "[", "]", "(", ")", ";", ",", ".", ":", "?", "#", "##", "<%", "%>", "<:", ":>",
"%:", "%:%:"

};

const char* operators[] = {

   "+", "-", "*", "/", "%", "++", "--", "==", "!=", ">", "<", ">=", "<=", "&&", "||",
"!",

   "&", "|", "^", "~", "<<", ">>", "=", "+=", "-=", "*=", "/=", "%=", "&=", "|=",
```


```
"^=", "<<=", ">>="
};
int isKeyword(const char* str) {
    for (int i = 0; i < sizeof(keywords) / sizeof(keywords[0]); i++) {
        if (strcmp(str, keywords[i]) == 0) {
            return 1;
        }
    }
    return 0;
}
int isPunctuator(char ch) {
    for (int i = 0; i < sizeof(punctuators) / sizeof(punctuators[0]); i++) {
        if (punctuators[i][0] == ch) {
            return 1;
        }
    }
    return 0;
}
int isOperator(const char* str) {
    for (int i = 0; i < sizeof(operators) / sizeof(operators[0]); i++) {
        if (strcmp(str, operators[i]) == 0) {
            return 1;
        }
    }
    return 0;
}
int isConstant(const char* str) {
    if (isdigit(str[0])) {
```


```
    for (int i = 1; i < strlen(str); i++) {
        if (!isdigit(str[i]) && str[i] != '.') {
            return 0;
        }
    }
    return 1;
  }
  return 0;
}
void tokenize(const char* code) {
  int i = 0;
  while (code[i] != '\0') {
    char token[MAX_TOKEN_SIZE] = {0};
    int j = 0;

    while (isspace(code[i])) {
      i++;
    }
    if (isalpha(code[i]) || code[i] == '_') {
      while (isalnum(code[i]) || code[i] == '_') {
        token[j++] = code[i++];
      }
      token[j] = '\0';
      if (isKeyword(token)) {
        printf("Keyword: %s\n", token);
      } else {
        printf("Identifier: %s\n", token);
      }
```


```
} else if (isdigit(code[i])) {
    while (isdigit(code[i])) {
        token[j++] = code[i++];
    }
    token[j] = '\0';
    printf("Constant: %s\n", token);
} else if (code[i] == '"' || code[i] == '\'') {
    char delimiter = code[i++];
    token[j++] = delimiter;
    while (code[i] != delimiter && code[i] != '\0') {
        token[j++] = code[i++];
    }
    if (code[i] == delimiter) {
        token[j++] = code[i++];
    }
    token[j] = '\0';
    printf("Literal: %s\n", token);
} else if (isPunctuator(code[i])) {
    token[j++] = code[i++];
    token[j] = '\0';
    printf("Punctuator: %s\n", token);
} else if (isOperator(&code[i])) {
    while (isOperator(&code[i])) {
        token[j++] = code[i++];
    }
    token[j] = '\0';
    printf("Operator: %s\n", token);
} else {
```

```
            i++;

        }

    }

}

int main() {

    char str[100];

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

    tokenize(str);

    return 0;

}
```

OUTPUT 1:

```
Enter a string: for(i<0;i<10;i++){scanf("%d",&a[i]);}
Keyword: for
Punctuator: (
Identifier: i
Punctuator: <
Constant: 0
Punctuator: ;
Identifier: i
Punctuator: <
Constant: 10
Punctuator: ;
Identifier: i
Punctuator: )
Punctuator: {
Keyword: scanf
Punctuator: (
Literal: "%d"
Punctuator: ,
Identifier: a
Punctuator: [
Identifier: i
Punctuator: ]
Punctuator: )
Punctuator: ;
Punctuator: }

Process returned 0 (0x0)   execution time : 31.172 s
Press ENTER to continue.
```

OUTPUT 2:

```
Enter a string: if(a>b){printf("a is greater");}else{("b is greater")};
Keyword: if
Punctuator: (
Identifier: a
Identifier: b
Punctuator: )
Punctuator: {
Keyword: printf
Punctuator: (
Literal: "a is greater"
Punctuator: )
Punctuator: ;
Punctuator: }
Keyword: else
Punctuator: {
Punctuator: (
Literal: "b is greater"
Punctuator: )
Punctuator: }
Punctuator: ;

Process returned 0 (0x0)   execution time : 45.577 s
Press ENTER to continue.
```

22BCE0682 Siddhant Bhagat

LAB TASK 2

CODE:

```cpp
#include <bits/stdc++.h>
using namespace std;

#define MAX_IDENTIFIERS 100

bool Invalid = false;
typedef struct {
    char name[50];
    int location;
    char type[10];
    int size;
} SymbolTableEntry;

int isValidIdentifier(char *identifier) {
    if (!isalpha(identifier[0]) && identifier[0] != '_') {
        return 0;
    }
    for (int i = 1; identifier[i]; i++) {
        if (!isalnum(identifier[i]) && identifier[i] != '_') {
            return 0;
        }
    }
    return 1;
}

int main() {
    SymbolTableEntry symbolTable[MAX_IDENTIFIERS];
    char input[256];
    int location = 1000;
    int index = 0;
    printf("Enter declaration statements: ");
    if (!fgets(input, sizeof(input), stdin)) {
        return 1; // Error reading input
    }
    char *token = strtok(input, " ,;\n");

    while (token != NULL) {
        if (strcmp(token, "int") == 0 || strcmp(token, "float") == 0 || strcmp(token, "char") == 0 || strcmp(token, "double") == 0) {
            char type[10];
            strcpy(type, token);
            int size = 0;
            if (strcmp(type, "int") == 0) size = 2;
            else if (strcmp(type, "float") == 0) size = 4;
            else if (strcmp(type, "double") == 0) size = 8;
            else if (strcmp(type, "char") == 0) size = 1;
            token = strtok(NULL, " ,;\n");

            while (token != NULL && (strcmp(token, "int") == 0 || strcmp(token, "float") == 0 || strcmp(token, "char") == 0 || strcmp(token, "double") == 0)) {
                if (isValidIdentifier(token)) {
                    strcpy(symbolTable[index].name, token);
                    symbolTable[index].location = location;
                    strcpy(symbolTable[index].type, type);
                    symbolTable[index].size = size;
                    location += size;
                    index++;
                } else {
                    cout << "Invalid identifier: " << token << endl;
                    Invalid = true;
                }
                token = strtok(NULL, " ,;\n");
            }
        } else {
            if (strlen(token) > 1) {
                cout << "Invalid identifier: " << token << endl;
                Invalid = true;
            }
            token = strtok(NULL, " ,;\n");
        }
    }

    if (!Invalid) {
        printf("\nSymbol Table:\n");
        printf("Location\tName\tType\tSize\n");
        for (int i = 0; i < index; i++) {
            printf("%d\t\t%s\t%s\t%d\n", symbolTable[i].location,
                   symbolTable[i].name, symbolTable[i].type, symbolTable[i].size);
        }
    }
    return 0;
}
```


```
#include <bits/stdc++.h>
using namespace std;

#define MAX_IDENTIFIERS 100
bool Invalid=false;
typedef struct {
    char
    name[50];int
    location; char
    type[10]; int
    size;
} SymbolTableEntry;

int isValidIdentifier(char *identifier) {
    if (!isalpha(identifier[0]) && identifier[0] != '_') {
        return 0;
    }
    for (int i = 1; i < strlen(identifier); i++) {
        if (!isalnum(identifier[i]) && identifier[i] != '_' && identifier[i] == ']' && identifier[i] == '[') {
            return 0;
        }
    }
    return 1;
}

int main() {
    SymbolTableEntry symbolTable[MAX_IDENTIFIERS];char
    input[256];
    int location = 1000;
    int index = 0;
    fgets(input, sizeof(input), stdin);
    char *token = strtok(input, "
    ,;\n");

    while (token != NULL) {
        if (strcmp(token, "int") == 0 || strcmp(token, "float") == 0 || strcmp(token, "char") == 0||
strcmp(token, "double") == 0) {
            char type[10];
            strcpy(type,
            token);int size = 0;
            int num =1;
            if(token!=NULL && token[1]=='['){int
                j=1;
                while(token[j]!=']' && token[j] != '\0') j++;
                if(token[j]==']'){
                    num = stoi(string(token).substr(1,j-1));
```

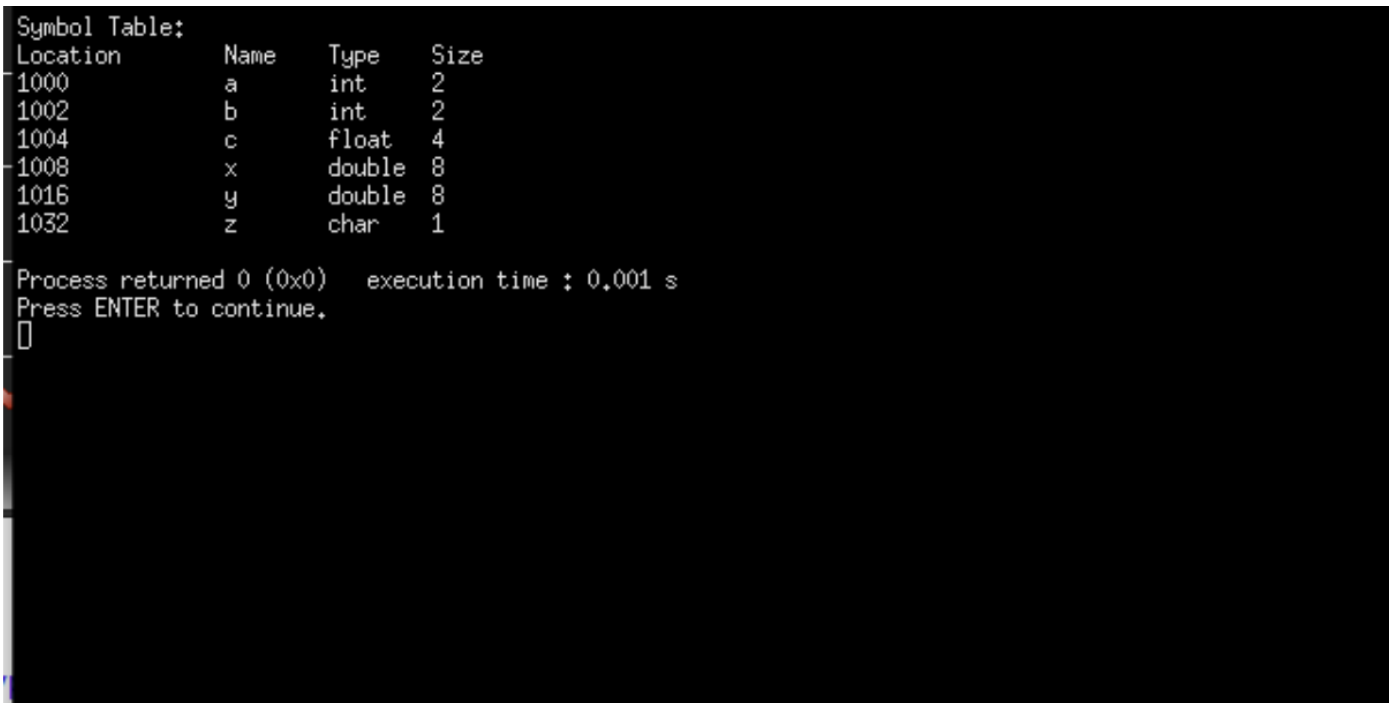

```
    }
}
if (strcmp(type, "int") == 0) size = num*2;
else if (strcmp(type, "float") == 0) size = num*4;
else if (strcmp(type, "double") == 0) size = num*8;
else if (strcmp(type, "char") == 0) size = num*1;
token = strtok(NULL, " ,;\n");
```

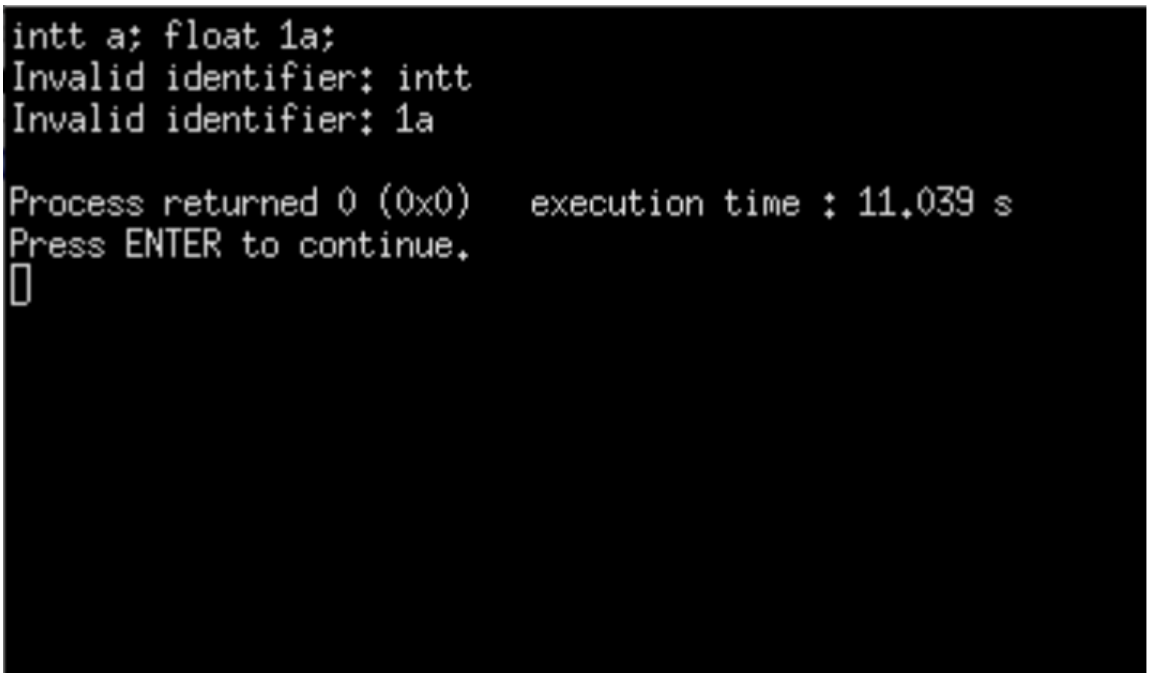

```
        while (token != NULL && (strcmp(token, "int") != 0 && strcmp(token, "float") != 0 &&
strcmp(token, "char") != 0 && strcmp(token, "double") != 0)) {
            if (isValidIdentifier(token)) {
                strcpy(symbolTable[index].name, token);
                symbolTable[index].location = location;
                strcpy(symbolTable[index].type, type);
                symbolTable[index].size = size;
                location += size;index++;
            }
            else{
                if (strlen(token)>1){
                cout<<"Invalid identifier: "<<token<<endl;
                Invalid=true;
            }
        }

            token = strtok(NULL, " ,;\n");
        }
        continue;
    }
     else{
        if (strlen(token)>1){
        cout<<"Invalid identifier: "<<token<<endl;
        Invalid=true;
        }
        }
    token = strtok(NULL, " ,;\n");
  }
  if(!Invalid){
    cout<<("\nSymbol Table:\n");
    cout<<("Name\tLocation\tSize\tType\n");

    for (int i = 0; i < index; i++) {
        printf("%s\t\t%d\t%d\t%s\n", symbolTable[i].name, symbolTable[i].location,
symbolTable[i].size, symbolTable[i].type);
    }
  }
  return 0;
```

Output:


Symbol Table:
Location Name Type Size
1000 a int 2
1002 b int 2
1004 c float 4
1008 x double 8
1016 y double 8
1032 z char 1
Process returned 0 (0x0) execution time : 0.001 s
Press ENTER to continue.


intt a; float 1a;
Invalid identifier: intt
Invalid identifier: 1a
Process returned 0 (0x0) execution time : 11.039 s
Press ENTER to continue.


```
int d[10];char a[10]

Symbol Table:
Name    Location        Size    Type
d[10]           1000    20      int
a[10]           1020    10      char

Process returned 0 (0x0)   execution time : 13.026 s
Press ENTER to continue.
```

```
intt a[10]
Invalid identifier: intt
Invalid identifier: a[10]

Process returned 0 (0x0)   execution time : 4.670 s
Press ENTER to continue.
```


VELLORE INSTITUTE OF TECHNOLOGY
VIT
உழைப்பே உயர்வு தரும்


VELLORE INSTITUTE OF TECHNOLOGY
VIT
உழைப்பே உயர்வு தரும்