**COMPILER DESIGN LAB COMMENT IDENTIFICATION**

**Name:** SreeDananjay S **Registration Number:** 21BAI1807 **Date:** 08-01-2024

**Slot:** L31+L32

**Course Code:** BCSE307P

**Programme:** Bachelor of Technology in Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning

**School:** School of Computer Science and Engineering(SCOPE)

**Q1) Write a program to identify whether a given line is a comment or not(Single line comment).**

**Code:**

#include <stdio.h> #include <stdlib.h> #include <string.h>

int main(int argc, char \*argv[])

{

char \*inputString;

inputString = malloc(1024 \* sizeof(char)); scanf("%[^\n]s", inputString);

inputString = realloc(inputString,strlen(inputString)+1);

int length = strlen(inputString);

// Performing checks if(length <2){

puts("The given input is not a Comment"); return EXIT\_FAILURE;

}

int startingIndex = 0;

while(startingIndex < length && inputString[startingIndex] == ' '){ startingIndex++;

}

if(startingIndex+1 >= length){

puts("The given input is not a Comment"); return EXIT\_FAILURE;

}

if(inputString[startingIndex] == '/' && inputString[startingIndex+1] == '/'){ printf("Comment found on index : %d\n", startingIndex);

}else{

puts("The given input is not a Comment");

}

return EXIT\_SUCCESS;

}

**Input and Output:**



**Q2) Write a program to identify whether a given line is a comment or not(Multi-line comment).**

**Code:**

#include <stdlib.h> #include <string.h>

int main(int argc, char \*argv[])

{

char \*inputString;

inputString = malloc(1024 \* sizeof(char)); scanf("%[^\n]s", inputString);

inputString = realloc(inputString,strlen(inputString)+1);

int length = strlen(inputString);

// Performing checks if(length <2){

puts("The given input is not a Comment"); return EXIT\_FAILURE;

}

int startingIndex = 0;

while(startingIndex < length && inputString[startingIndex] == ' '){ startingIndex++;

}

if(startingIndex+1 >= length){

puts("The given input is not a Comment"); return EXIT\_FAILURE;

}

if(inputString[startingIndex] != '/' || inputString[startingIndex+1] != '\*'){ puts("The given input is not a comment");

return EXIT\_FAILURE;

}

if(inputString[length-2] != '\*' || inputString[length-1] != '/'){ puts(The give input is not a comment");

return EXIT\_FAILURE;

}

printf("The given line has comment opened at index : %d", startingIndex);

return EXIT\_SUCCESS;

}

**Input and Output:**



**Q3) Finding the comment (Single and Multi-line) in a C program that is given as input.**

**Code:**

#include <stdio.h> #include <string.h>

#define MAX\_LINES 1000

#define MAX\_LINE\_LENGTH 100

// Function to check if a line contains a single-line comment int hasSingleLineComment(const char \*line) {

return (strstr(line, "//") != NULL);

}

// Function to check if a line contains the start of a multi-line comment int hasMultiLineCommentStart(const char \*line) {

return (strstr(line, "/\*") != NULL);

}

// Function to check if a line contains the end of a multi-line comment int hasMultiLineCommentEnd(const char \*line) {

return (strstr(line, "\*/") != NULL);

}

int main() {

char input[MAX\_LINES][MAX\_LINE\_LENGTH]; int lineCount = 0;

// Get multiline input until an empty line is entered

printf("Enter multiline input. Press Enter on an empty line to finish:\n");

do {

// Read a line of input

fgets(input[lineCount], sizeof(input[0]), stdin);

// Check if the line is empty (contains only '\n') if (input[lineCount][0] == '\n') {

break; // Exit the loop if an empty line is encountered

}

lineCount++;

} while (lineCount < MAX\_LINES); // Limit the number of lines for simplicity

// Process the multiline input printf("Analysis of multiline input:\n");

for (int i = 0; i < lineCount; i++) {

// Check for single-line comment if(hasSingleLineComment(input[i])) {

printf("Line %d: single-line comment\n", i + 1);

}

else if(hasMultiLineCommentStart(input[i])) {

// Check if the multi-line comment ends on the same line if(hasMultiLineCommentEnd(input[i])) {

printf("Line %d: multi-line comment\n", i + 1);

}

else {

printf("Line %d: multi-line comment (not terminated)\n", i + 1);

}

}

else {

printf("Line %d: not a comment\n", i + 1);

}

}

}

**Input and Output:**

