**PRACTICAL - 4**

|  |  |
| --- | --- |
|  | **Problem statement**  Program to count the number of characters, words, spaces and lines in a given input file. USING Lex |

**SOLUTION:**

|  |  |  |
| --- | --- | --- |
| **CODE:**   |  | | --- | | #include <stdio.h>  #include <stdlib.h>  int main() {  int characters = 0, words = 0, lines = 0, spaces = 0;  char ch;  FILE \*file;  // Open file, prompt for file name  char filename[100];  printf("Enter the file name (with path if necessary): ");  scanf("%s", filename);  file = fopen(filename, "r");  if (file == NULL) {  printf("Unable to open the file.\n");  exit(EXIT\_FAILURE);  }  // Read file character by character  while ((ch = fgetc(file)) != EOF) {  characters++; // Increment character count  if (ch == ' ') {  spaces++; // Increment space count  }  if (ch == '\n') {  lines++; // Increment line count  }  if (ch == ' ' || ch == '\n' || ch == '\t') {  words++; // Increment word count for boundaries  }  }  // Check if the last character read is not a boundary  if (characters > 0) {  words++; // Increment for the last word if necessary  }  // Print results  printf("Total characters = %d\n", characters);  printf("Total words = %d\n", words);  printf("Total spaces = %d\n", spaces);  printf("Total lines = %d\n", lines);  // Close the file  fclose(file);  return 0;  } |   **OUTPUT:**   |  | | --- | |  | |

|  |  |
| --- | --- |
| **2.** | **Problem statement**  Program to count the numbers of comment lines in a given C program. Also Eliminate Them and copy the resulting program into a separate file |

**SOLUTION:**

|  |  |  |
| --- | --- | --- |
| **CODE:**   |  | | --- | | #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #define MAX\_LINE\_LENGTH 1024  void count\_and\_eliminate\_comments(const char \*input\_file, const char \*output\_file) {  FILE \*source\_file, \*dest\_file;  char line[MAX\_LINE\_LENGTH];  int comment\_count = 0;  // Open the source file for reading  source\_file = fopen(input\_file, "r");  if (source\_file == NULL) {  printf("Unable to open the source file.\n");  exit(EXIT\_FAILURE);  }  // Open the destination file for writing  dest\_file = fopen(output\_file, "w");  if (dest\_file == NULL) {  fclose(source\_file);  printf("Unable to create the destination file.\n");  exit(EXIT\_FAILURE);  }  while (fgets(line, sizeof(line), source\_file)) {  // Check for single-line comments  char \*single\_line\_comment = strstr(line, "//");  char \*multi\_line\_comment\_start = strstr(line, "/\*");  // Count single-line comments  if (single\_line\_comment) {  comment\_count++;  \*single\_line\_comment = '\0'; // Eliminate single-line comment  }  // Handle multi-line comments  char \*multi\_line\_comment\_end = strstr(line, "\*/");  if (multi\_line\_comment\_start) {  comment\_count++;  if (multi\_line\_comment\_end) {  \*multi\_line\_comment\_start = '\0'; // Eliminate start of multi-line comment  multi\_line\_comment\_end += 2; // Move past the end comment  strcpy(line, multi\_line\_comment\_end); // Copy the rest of the line  } else {  // If we found a start but no end, we need to ignore this line entirely  continue;  }  }  // Write the line without comments to the destination file  fputs(line, dest\_file);  }  // Closing files  fclose(source\_file);  fclose(dest\_file);  // Print the number of comment lines found  printf("Total comment lines found: %d\n", comment\_count);  }  int main() {  const char \*input\_file = "input.c"; // Specify your input C file  const char \*output\_file = "output.c"; // Specify your output C file  count\_and\_eliminate\_comments(input\_file, output\_file);  return 0;  } |   **OUTPUT:**   |  | | --- | |  | |

|  |  |
| --- | --- |
| **3.** | **Problem statement**  Program to recognize a valid arithmetic expression and to recognize the identifiers and operators present . Print them separately. |

**SOLUTION:**

|  |  |  |
| --- | --- | --- |
| **CODE:**   |  | | --- | | #include <stdio.h>  #include <string.h>  #include <ctype.h>  int main() {  char expression[100];  char token[100];  int i = 0, j = 0, isIdentifier = 0, isOperator = 0, isValid = 1;  // Get the arithmetic expression from the user  printf("Enter an arithmetic expression: ");  scanf("%s", expression);  // Tokenize the expression  while (expression[i] != '\0') {  if (isalnum(expression[i]) || expression[i] == '\_') {  // If it's a letter, digit, or underscore, it's part of an identifier  token[j] = expression[i];  isIdentifier = 1;  isOperator = 0;  } else {  // If it's an operator, reset the token and set the flag  if (isIdentifier) {  token[j] = '\0';  printf("Identifier: %s\n", token);  j = 0;  isIdentifier = 0;  }  if (expression[i] == '+' || expression[i] == '-' || expression[i] == '\*' || expression[i] == '/' || expression[i] == '(' || expression[i] == ')') {  token[j] = expression[i];  token[j + 1] = '\0';  printf("Operator: %s\n", token);  isOperator = 1;  } else {  // Invalid character, mark expression as invalid  isValid = 0;  break;  }  j = 0;  }  i++;  j++;  }  // Handle the last token  if (isIdentifier) {  token[j] = '\0';  printf("Identifier: %s\n", token);  }  // Print the validation result  if (isValid) {  printf("Expression is valid.\n");  } else {  printf("Expression is invalid.\n");  }  return 0;  } |   **OUTPUT:**   |  | | --- | |  | |

|  |  |
| --- | --- |
| **4.** | **Problem statement**  Program to recognize whether a given sentence is a simple or compound. |

**SOLUTION:**

|  |  |  |
| --- | --- | --- |
| **CODE:**   |  | | --- | | #include <stdio.h>  #include <string.h>  #include <ctype.h>  int main() {  char sentence[1000];  char \*token;  int isCompound = 0;  // Get the sentence from the user  printf("Enter a sentence: ");  fgets(sentence, sizeof(sentence), stdin);  // Remove trailing newline character  sentence[strcspn(sentence, "\n")] = 0;  // Tokenize the sentence based on spaces  token = strtok(sentence, " ");  while (token != NULL) {  // Check for common conjunctions (and, but, or, so, for, nor, yet)  if (strcmp(token, "and") == 0 || strcmp(token, "but") == 0 ||  strcmp(token, "or") == 0 || strcmp(token, "so") == 0 ||  strcmp(token, "for") == 0 || strcmp(token, "nor") == 0 ||  strcmp(token, "yet") == 0) {  isCompound = 1;  break;  }  token = strtok(NULL, " ");  }  // Print the result  if (isCompound) {  printf("The sentence is compound.\n");  } else {  printf("The sentence is simple.\n");  }  return 0;  } |   **OUTPUT:**   |  | | --- | |  | |

|  |  |
| --- | --- |
| **5.** | **Problem statement**  Program to recognize and count the number identifier in input file. |

**SOLUTION:**

|  |  |  |
| --- | --- | --- |
| **CODE:**   |  | | --- | | #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <ctype.h>  #define MAX\_WORD\_LEN 100  #define MAX\_LINE\_LEN 256  // Function to check if a word is a valid identifier  int is\_identifier(const char \*word) {  if (!isalpha(word[0]) && word[0] != '\_') { // First character must be a letter or underscore  return 0; // Not a valid identifier  }  int i;  for (i = 1; word[i] != '\0'; i++) { // Check subsequent characters  if (!isalnum(word[i]) && word[i] != '\_') {  return 0; // Contains invalid characters  }  }  return 1; // Valid identifier  }  // Function to count identifiers in a file  int count\_identifiers(const char \*file\_name) {  FILE \*file = fopen(file\_name, "r");  if (file == NULL) {  perror("Error opening file");  return -1; // Error opening file  }  char line[MAX\_LINE\_LEN];  char word[MAX\_WORD\_LEN];  int count = 0;  // Read lines from file  while (fgets(line, sizeof(line), file)) {  char \*token = strtok(line, " \t\n"); // Split line into words  while (token != NULL) {  if (is\_identifier(token)) {  printf("%s is an identifier\n", token);  count++;  } else if (isdigit(token[0]) || (token[0] == '-' && isdigit(token[1]))) {  printf("%s is a number\n", token);  }  token = strtok(NULL, " \t\n"); // Get next token  }  }  fclose(file); // Close the file  return count; // Return total count of identifiers  }  int main() {  char file\_name[100];  printf("Enter the filename: ");  fgets(file\_name, sizeof(file\_name), stdin); // Read the filename from input  file\_name[strcspn(file\_name, "\n")] = '\0'; // Remove the newline character  int total\_identifiers = count\_identifiers(file\_name);  if (total\_identifiers >= 0) {  printf("Total identifiers are: %d\n", total\_identifiers); // Print total identifiers  }  return 0;  } |   **OUTPUT:**   |  | | --- | |  | |