EXPERIMENT-3

Program:

To implement a c program for the characterization of a file and removal of the comments in that file.

Theory:

In the context of programming or file analysis, "characterization" refers to the process of analysing and describing certain features or properties of a given entity. In the case of a file or program, characterization typically involves examining various aspects of its content to gain insights into its structure, composition, or specific elements.

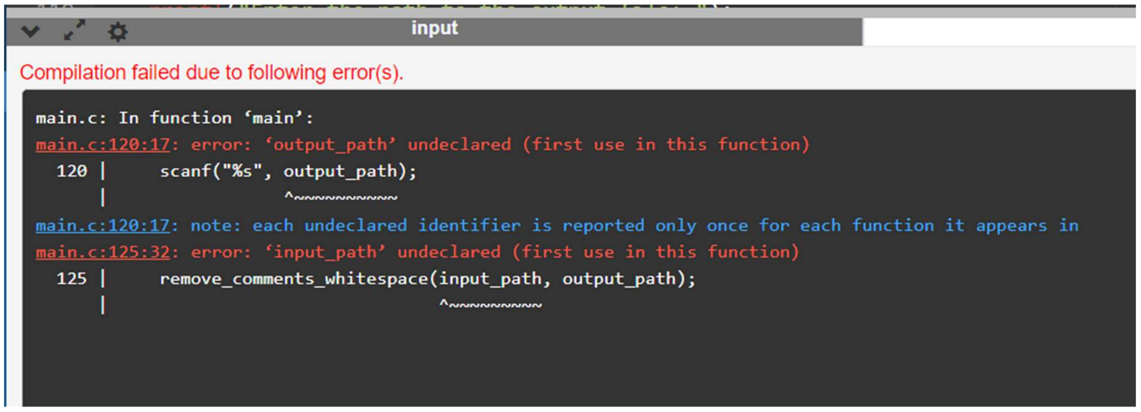
To implement a C program for the characterization of a file and removal of comments, you can follow these general steps:

1. **File Characterization:**
    - Open the input file in read mode.
    - Read each line of the file.
    - For each line, check if it contains comments or functions.
    - Keep track of the number of lines, comments, and functions.
2. **Remove Comments and Whitespaces:**
    - Open the input file again in read mode.
    - Open a new output file in write mode.
    - Read each line of the input file.
    - For each line, remove comments (both single-line and multi-line).
    - Remove leading and trailing whitespaces.
    - Write the modified line to the output file.

Input:

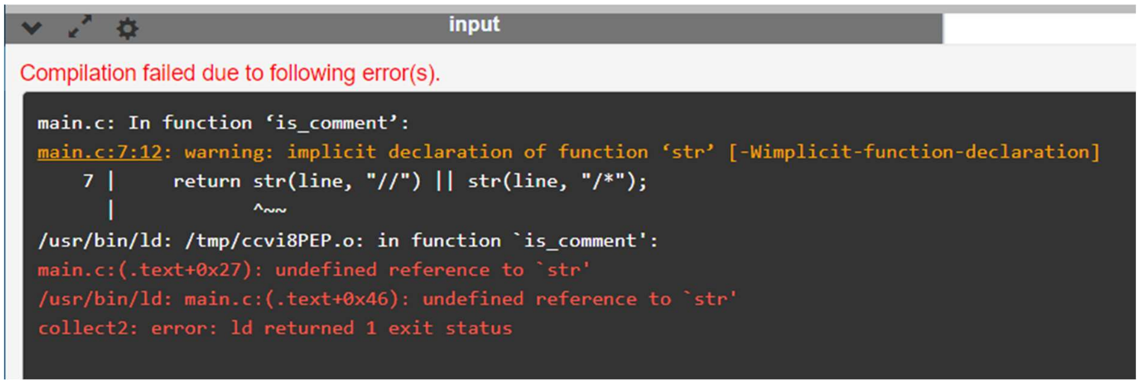
```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_LINE_SIZE 1000

int is_comment(const char *line) {

  return strstr(line, "//") || strstr(line, "/*");}

int is_function(const char *line) {
  return (strstr(line, "(") != NULL) && (strstr(line, ")") != NULL);}
void create_input_file(const char *file_path) {
  FILE *file = fopen(file_path, "w");
  if (file == NULL) {
      perror("Error creating file");
      exit(EXIT_FAILURE);  }
  printf("Enter your C program (type 'exit' on a new line to finish):\n");
  char line[MAX_LINE_SIZE];
  while (1)   {
      fgets(line, MAX_LINE_SIZE, stdin);
      if (strcmp(line, "exit\n") == 0)        {
          break;       }
      fputs(line, file);    }
  fclose(file);}
void characterize_program(const char *file_path) {
  FILE *file = fopen(file_path, "r");
  if (file == NULL)     {
      perror("Error opening file");
      exit(EXIT_FAILURE);    }
  char line[MAX_LINE_SIZE];
  int num_lines = 0;
  int num_comments = 0;
  int num_functions = 0;
  while (fgets(line, MAX_LINE_SIZE, file) != NULL)     {
      num_lines++;
      if (is_comment(line))        {
          num_comments++;       }
      if (is_function(line))        {
          num_functions++;       }    }
  fclose(file);
  printf("Number of lines: %d\n", num_lines);
  printf("Number of comments: %d\n", num_comments);
  printf("Number of functions: %d\n", num_functions);}
void remove_comments_whitespace(const char *input_path, const char *output_path) {
  FILE *input_file = fopen(input_path, "r");
  FILE *output_file = fopen(output_path, "w");
  if (input_file == NULL || output_file == NULL)     {
      perror("Error opening files");
      exit(EXIT_FAILURE);    }
  char line[MAX_LINE_SIZE];
  while (fgets(line, MAX_LINE_SIZE, input_file) != NULL)     {
      char *comment_start = strstr(line, "//");
      if (comment_start != NULL)       {
          *comment_start = '\0';       }
      comment_start = strstr(line, "/*");
      char *comment_end = strstr(line, "*/");
      if (comment_start != NULL && comment_end != NULL)       {
          memmove(comment_start, comment_end + 2, strlen(comment_end + 2) + 1);
      } else if (comment_start != NULL)      {
          *comment_start = '\0';      }
      int i = 0, j = strlen(line) - 1;
      while (i <= j && (line[i] == ' ' || line[i] == '\t' || line[i] == '\n'))        {
          i++;        }
      while (j >= i && (line[j] == ' ' || line[j] == '\t' || line[j] == '\n'))        {
          j--;        }
      fprintf(output_file, "%.*s\n", j - i + 1, line + i);    }
  fclose(input_file);
  fclose(output_file);}
int main() {
  char input_path[MAX_LINE_SIZE];
  char output_path[MAX_LINE_SIZE];
  printf("Enter the path to the input file: ");
  scanf("%s", input_path);
  create_input_file(input_path);
  printf("Input file created successfully!\n");
  printf("Enter the path to the output file: ");
  scanf("%s", output_path);
  printf("\nCharacterizing the input file:\n");
  characterize_program(input_path);
  remove_comments_whitespace(input_path, output_path);
  printf("Comments and whitespaces removed successfully!\n");
  return 0;
}
```

Output:



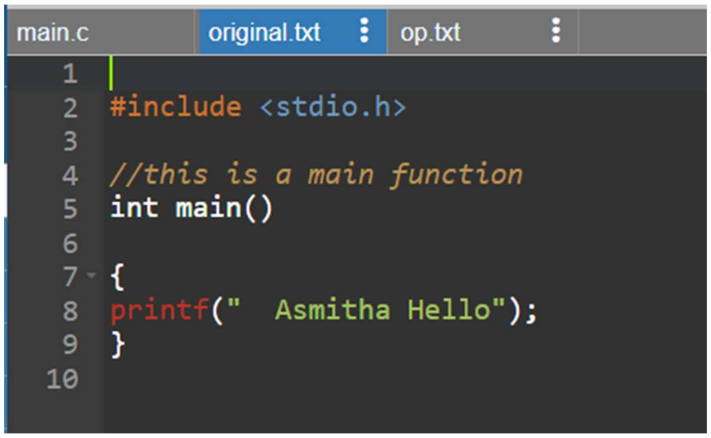
Compilation failed due to following error(s).

```
main.c: In function ‘main’:
main.c:120:17: error: ‘output_path’ undeclared (first use in this function)
  120 |     scanf("%s", output_path);
      |                 ^~~~~~~~~~~
main.c:120:17: note: each undeclared identifier is reported only once for each function it appears in
main.c:125:32: error: ‘input_path’ undeclared (first use in this function)
  125 |     remove_comments_whitespace(input_path, output_path);
      |                                ^~~~~~~~~~
```

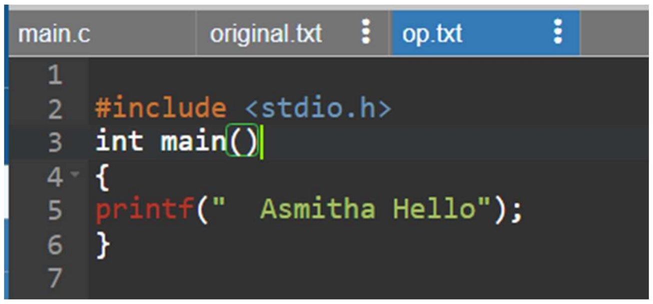


Compilation failed due to following error(s).

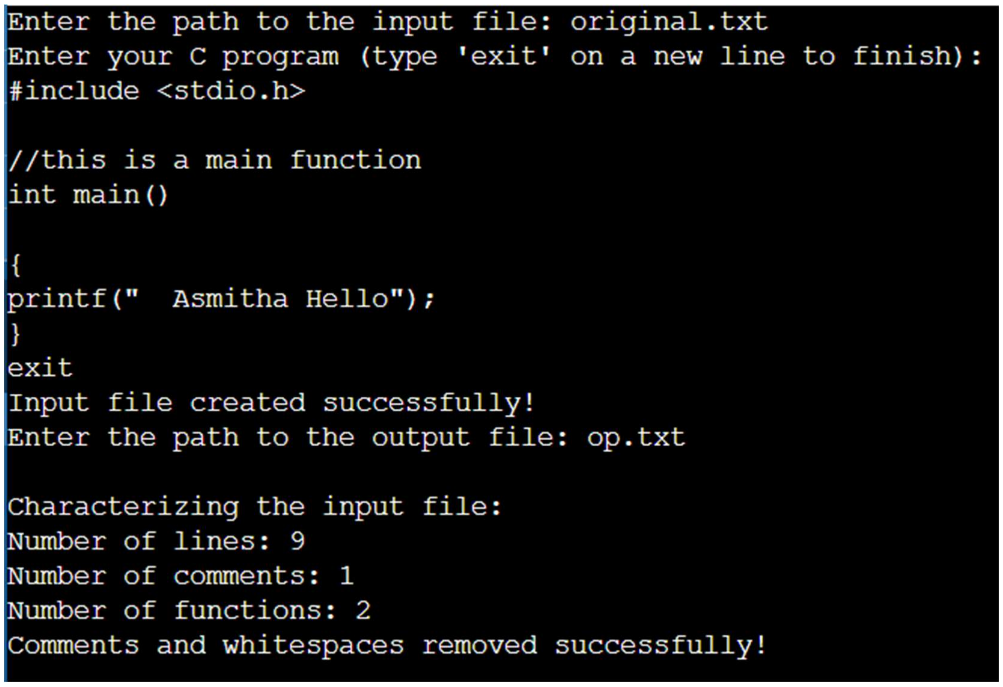
```
main.c: In function ‘is_comment’:
main.c:7:12: warning: implicit declaration of function ‘str’ [-Wimplicit-function-declaration]
    7 |     return str(line, "//") || str(line, "/*");
      |            ^~~
/usr/bin/ld: /tmp/ccvi8PEP.o: in function `is_comment':
main.c:(.text+0x27): undefined reference to `str'
/usr/bin/ld: main.c:(.text+0x46): undefined reference to `str'
collect2: error: ld returned 1 exit status
```



```c

#include <stdio.h>

//this is a main function
int main()

{
printf("  Asmitha Hello");
}

```



```c

#include <stdio.h>
int main()
{
printf("  Asmitha Hello");
}

```



```
Enter the path to the input file: original.txt
Enter your C program (type 'exit' on a new line to finish):
#include <stdio.h>

//this is a main function
int main()

{
printf("  Asmitha Hello");
}
exit
Input file created successfully!
Enter the path to the output file: op.txt

Characterizing the input file:
Number of lines: 9
Number of comments: 1
Number of functions: 2
Comments and whitespaces removed successfully!
```