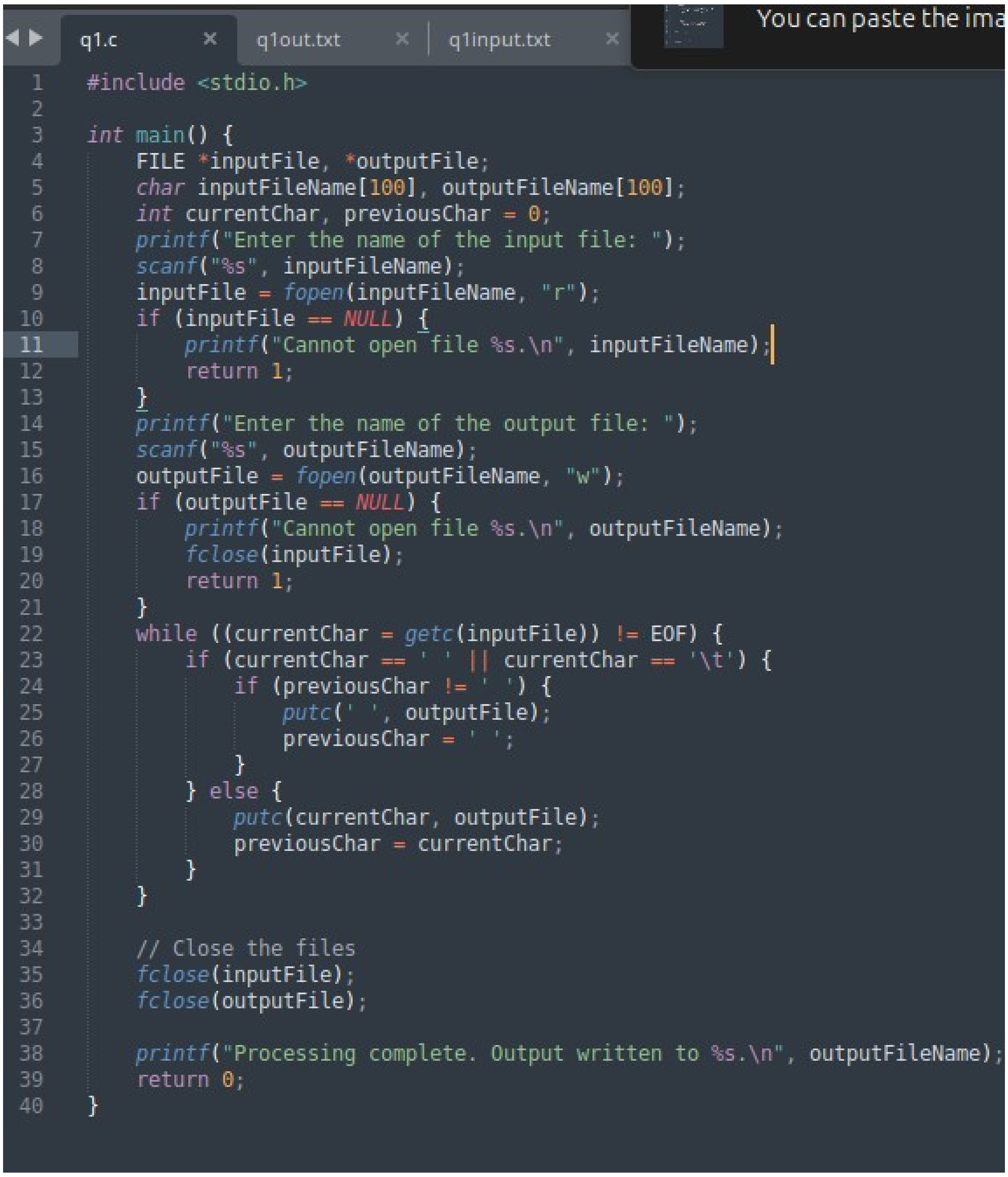
NAME:AYANIKA PAUL
Roll No. 22
D1

LAB 2

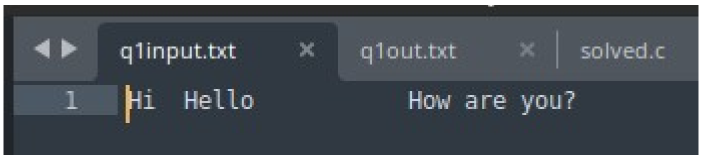
Q1)Write a ‘C’ program
1. That takes a file as input and replaces blank spaces and tabs by single space and writes the output to a file.

CODE:



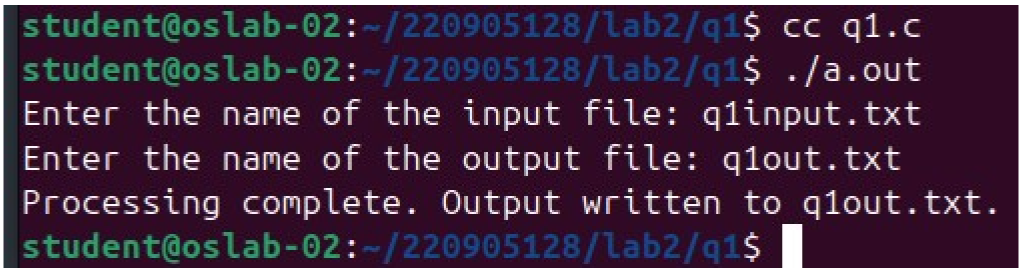
```c
#include <stdio.h>

int main() {
    FILE *inputFile, *outputFile;
    char inputFileName[100], outputFileName[100];
    int currentChar, previousChar = 0;
    printf("Enter the name of the input file: ");
    scanf("%s", inputFileName);
    inputFile = fopen(inputFileName, "r");
    if (inputFile == NULL) {
        printf("Cannot open file %s.\n", inputFileName);
        return 1;
    }
    printf("Enter the name of the output file: ");
    scanf("%s", outputFileName);
    outputFile = fopen(outputFileName, "w");
    if (outputFile == NULL) {
        printf("Cannot open file %s.\n", outputFileName);
        fclose(inputFile);
        return 1;
    }
    while ((currentChar = getc(inputFile)) != EOF) {
        if (currentChar == ' ' || currentChar == '\t') {
            if (previousChar != ' ') {
                putc(' ', outputFile);
                previousChar = ' ';
            }
        } else {
            putc(currentChar, outputFile);
            previousChar = currentChar;
        }
    }

    // Close the files
    fclose(inputFile);
    fclose(outputFile);

    printf("Processing complete. Output written to %s.\n", outputFileName);
    return 0;
}
```

Input file : q1in.txt



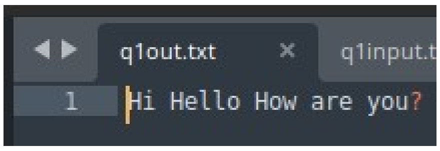
```
Hi  Hello           How are you?
```

Terminal



```
student@oslab-02:~/220905128/lab2/q1$ cc q1.c
student@oslab-02:~/220905128/lab2/q1$ ./a.out
Enter the name of the input file: q1input.txt
Enter the name of the output file: q1out.txt
Processing complete. Output written to q1out.txt.
student@oslab-02:~/220905128/lab2/q1$ 
```

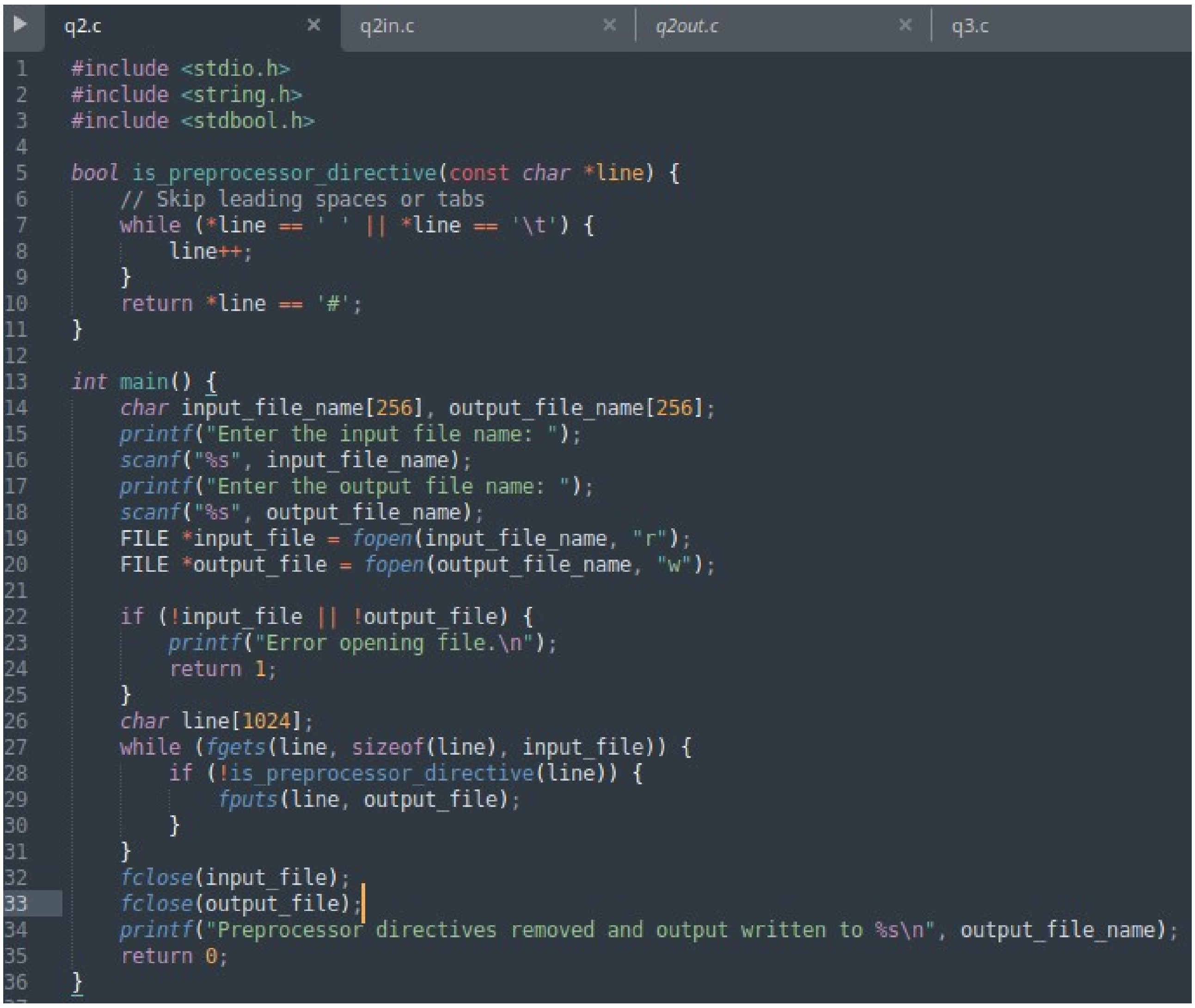
Output file: q1out.txt



```
Hi Hello How are you?
```

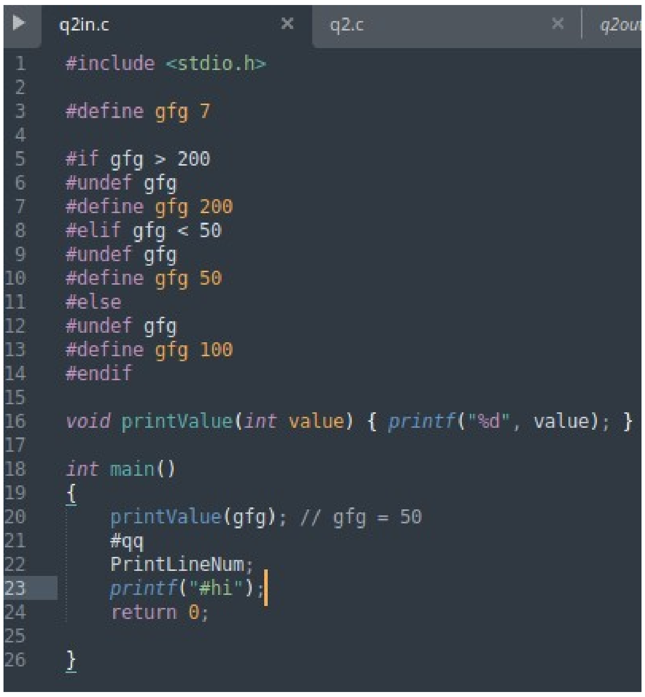
Q2) To discard preprocessor directives from the given input ‘C’ file.

CODE:



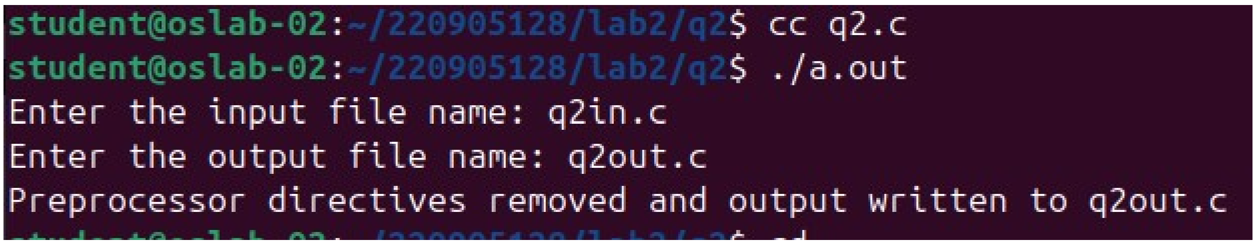
```c
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

bool is_preprocessor_directive(const char *line) {
    // Skip leading spaces or tabs
    while (*line == ' ' || *line == '\t') {
        line++;
    }
    return *line == '#';
}

int main() {
    char input_file_name[256], output_file_name[256];
    printf("Enter the input file name: ");
    scanf("%s", input_file_name);
    printf("Enter the output file name: ");
    scanf("%s", output_file_name);
    FILE *input_file = fopen(input_file_name, "r");
    FILE *output_file = fopen(output_file_name, "w");

    if (!input_file || !output_file) {
        printf("Error opening file.\n");
        return 1;
    }
    char line[1024];
    while (fgets(line, sizeof(line), input_file)) {
        if (!is_preprocessor_directive(line)) {
            fputs(line, output_file);
        }
    }
    fclose(input_file);
    fclose(output_file);
    printf("Preprocessor directives removed and output written to %s\n", output_file_name);
    return 0;
}
```

Input file: q2in.c



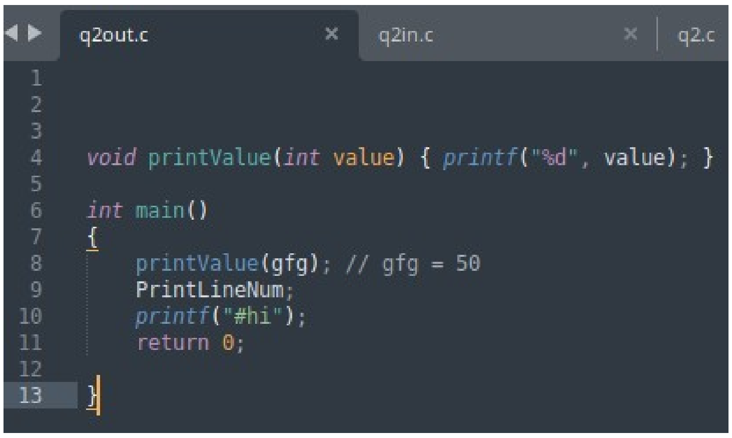
```c
#include <stdio.h>

#define gfg 7

#if gfg > 200
#undef gfg
#define gfg 200
#elif gfg < 50
#undef gfg
#define gfg 50
#else
#undef gfg
#define gfg 100
#endif

void printValue(int value) { printf("%d", value); }

int main()
{
    printValue(gfg); // gfg = 50
    #qq
    PrintLineNum;
    printf("#hi");
    return 0;

}
```

Terminal:



```
student@oslab-02:~/220905128/lab2/q2$ cc q2.c
student@oslab-02:~/220905128/lab2/q2$ ./a.out
Enter the input file name: q2in.c
Enter the output file name: q2out.c
Preprocessor directives removed and output written to q2out.c
```

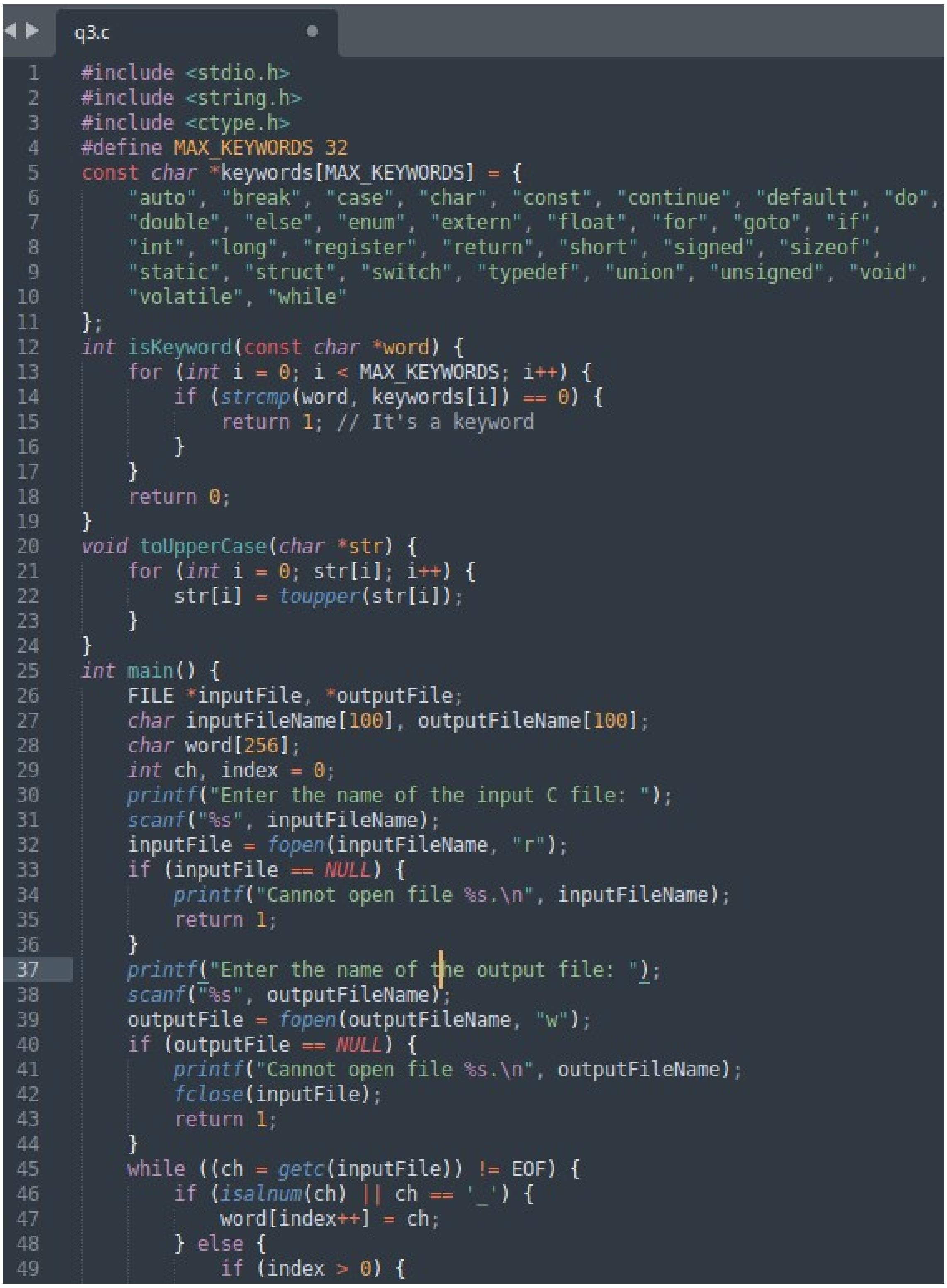
Output file: q2out.c



```c



void printValue(int value) { printf("%d", value); }

int main()
{
    printValue(gfg); // gfg = 50
    PrintLineNum;
    printf("#hi");
    return 0;

}
```

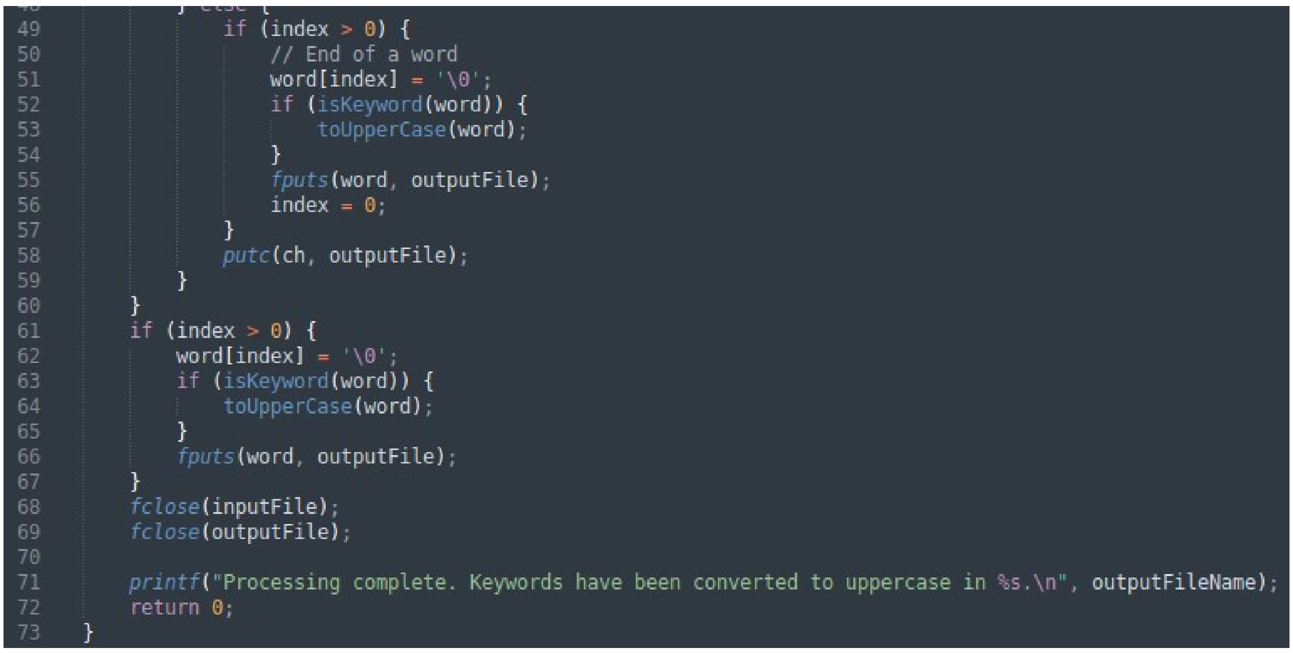
Q3)That takes C program as input, recognizes all the keywords and prints them in upper case.

CODE:



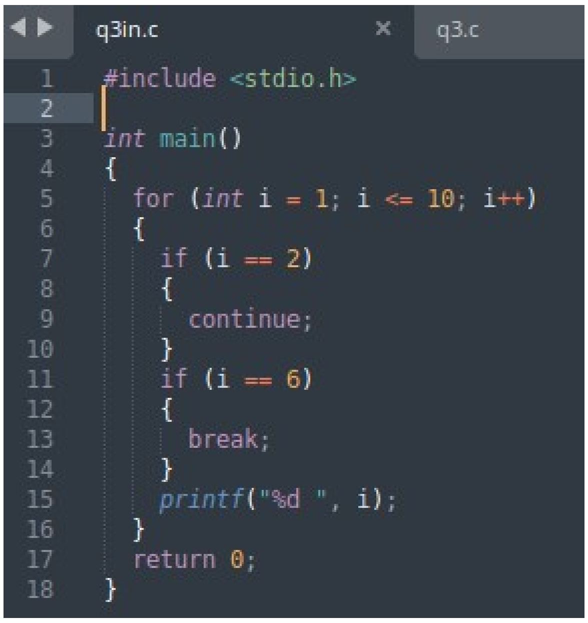
q3.c

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define MAX_KEYWORDS 32
const char *keywords[MAX_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"
};
int isKeyword(const char *word) {
    for (int i = 0; i < MAX_KEYWORDS; i++) {
        if (strcmp(word, keywords[i]) == 0) {
            return 1; // It's a keyword
        }
    }
    return 0;
}
void toUpperCase(char *str) {
    for (int i = 0; str[i]; i++) {
        str[i] = toupper(str[i]);
    }
}
int main() {
    FILE *inputFile, *outputFile;
    char inputFileName[100], outputFileName[100];
    char word[256];
    int ch, index = 0;
    printf("Enter the name of the input C file: ");
    scanf("%s", inputFileName);
    inputFile = fopen(inputFileName, "r");
    if (inputFile == NULL) {
        printf("Cannot open file %s.\n", inputFileName);
        return 1;
    }
    printf("Enter the name of the output file: ");
    scanf("%s", outputFileName);
    outputFile = fopen(outputFileName, "w");
    if (outputFile == NULL) {
        printf("Cannot open file %s.\n", outputFileName);
        fclose(inputFile);
        return 1;
    }
    while ((ch = getc(inputFile)) != EOF) {
        if (isalnum(ch) || ch == '_') {
            word[index++] = ch;
        } else {
            if (index > 0) {
```



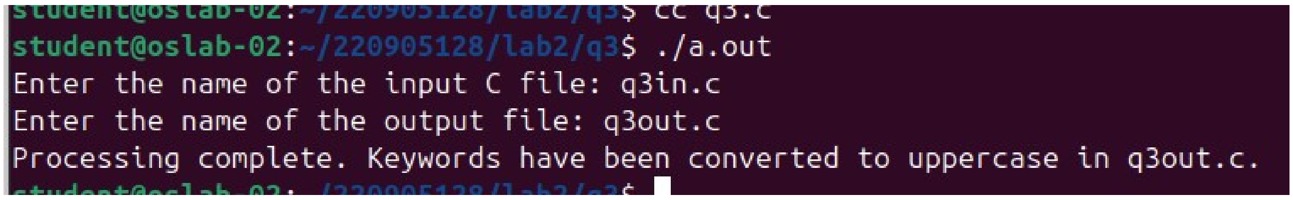
```c
            if (index > 0) {
                // End of a word
                word[index] = '\0';
                if (isKeyword(word)) {
                    toUpperCase(word);
                }
                fputs(word, outputFile);
                index = 0;
            }
            putc(ch, outputFile);
        }
    }
    if (index > 0) {
        word[index] = '\0';
        if (isKeyword(word)) {
            toUpperCase(word);
        }
        fputs(word, outputFile);
    }
    fclose(inputFile);
    fclose(outputFile);

    printf("Processing complete. Keywords have been converted to uppercase in %s.\n", outputFileName);
    return 0;
}
```

Input file: q3in.c



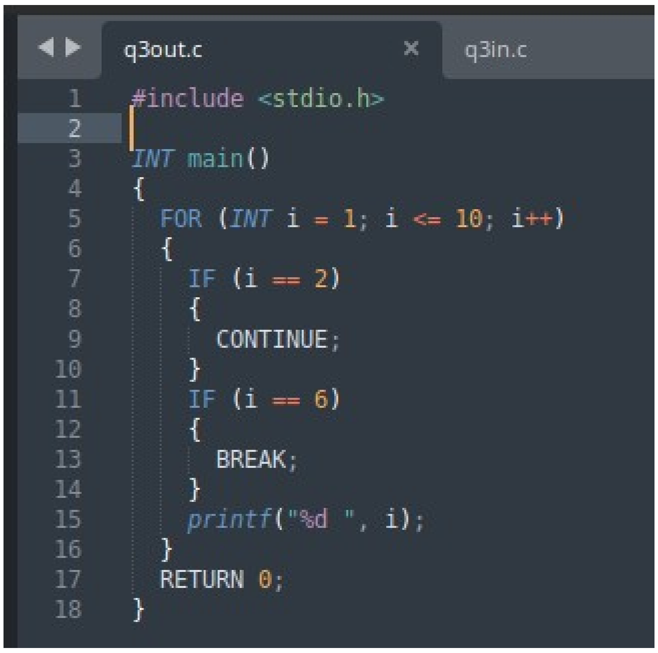
```c
#include <stdio.h>

int main()
{
  for (int i = 1; i <= 10; i++)
  {
    if (i == 2)
    {
      continue;
    }
    if (i == 6)
    {
      break;
    }
    printf("%d ", i);
  }
  return 0;
}
```

Terminal:



```
student@oslab-02:~/220905128/lab2/q3$ cc q3.c
student@oslab-02:~/220905128/lab2/q3$ ./a.out
Enter the name of the input C file: q3in.c
Enter the name of the output file: q3out.c
Processing complete. Keywords have been converted to uppercase in q3out.c.
student@oslab-02:~/220905128/lab2/q3$
```

Output file: q3out.c

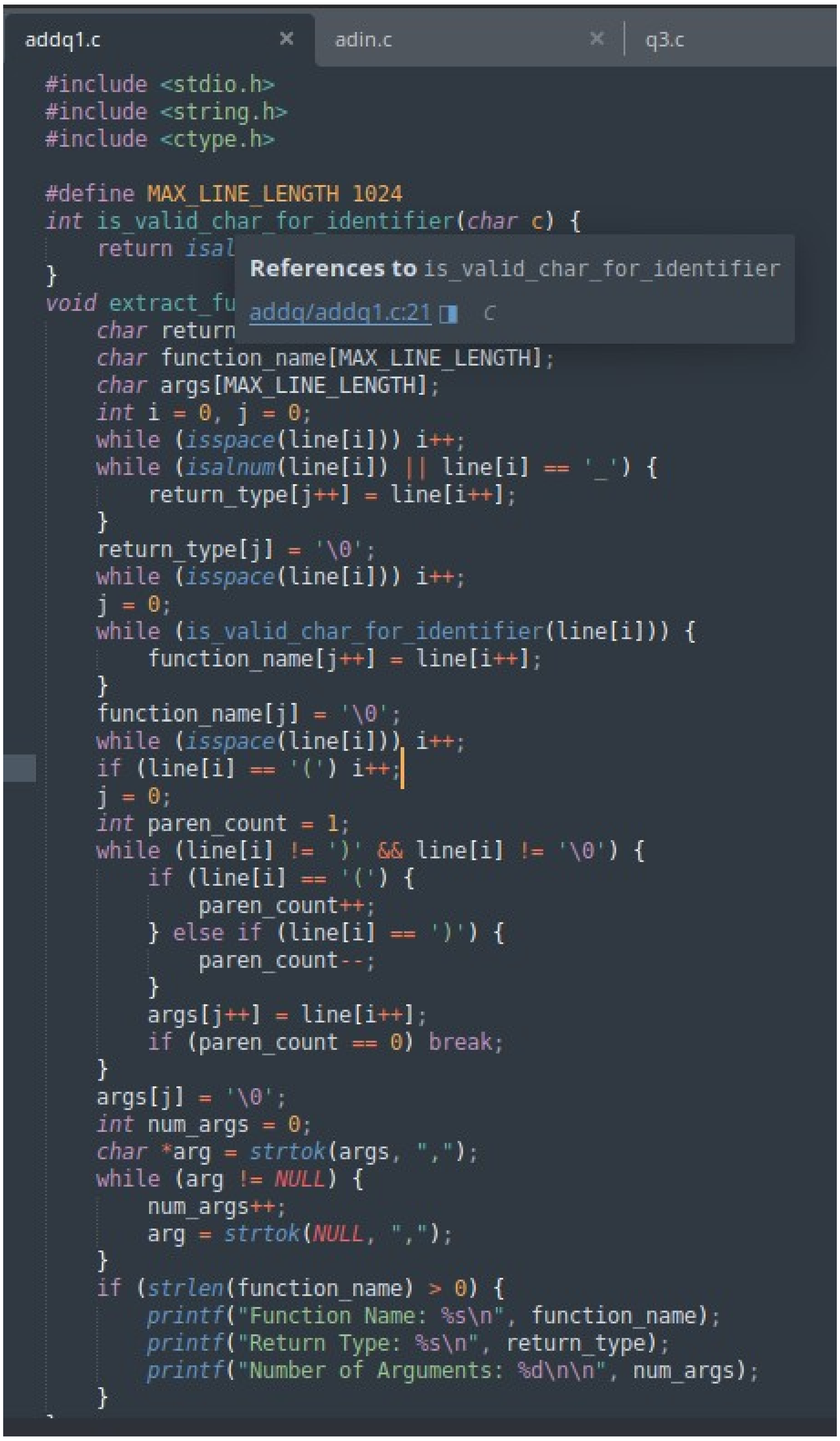


```c
#include <stdio.h>

INT main()
{
  FOR (INT i = 1; i <= 10; i++)
  {
    IF (i == 2)
    {
      CONTINUE;
    }
    IF (i == 6)
    {
      BREAK;
    }
    printf("%d ", i);
  }
  RETURN 0;
}
```

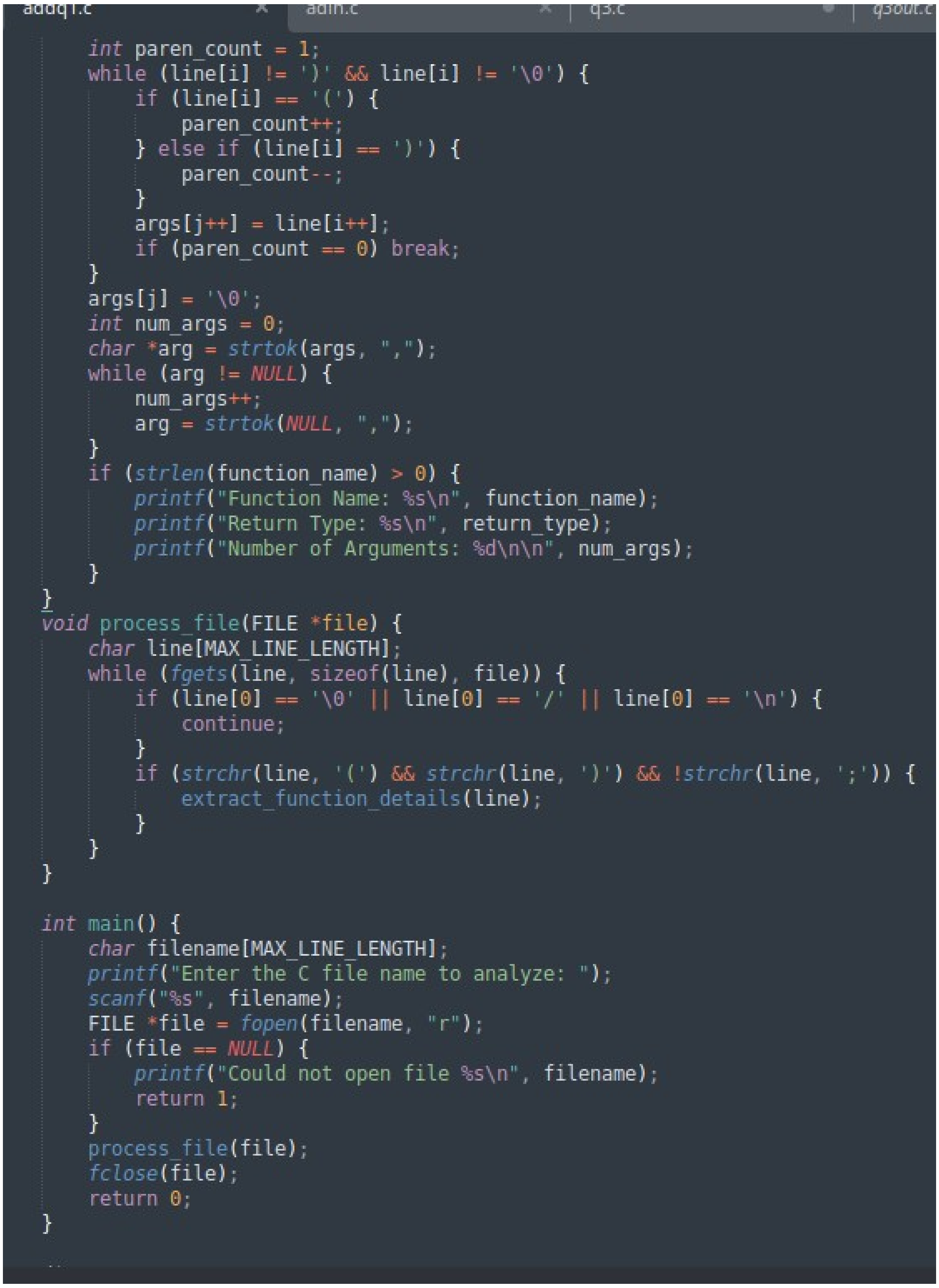
ADDITIONAL Q

Q1)Write a program to display the function names present in the given input ‘C’ file along with its return type and number of arguments.

CODE:

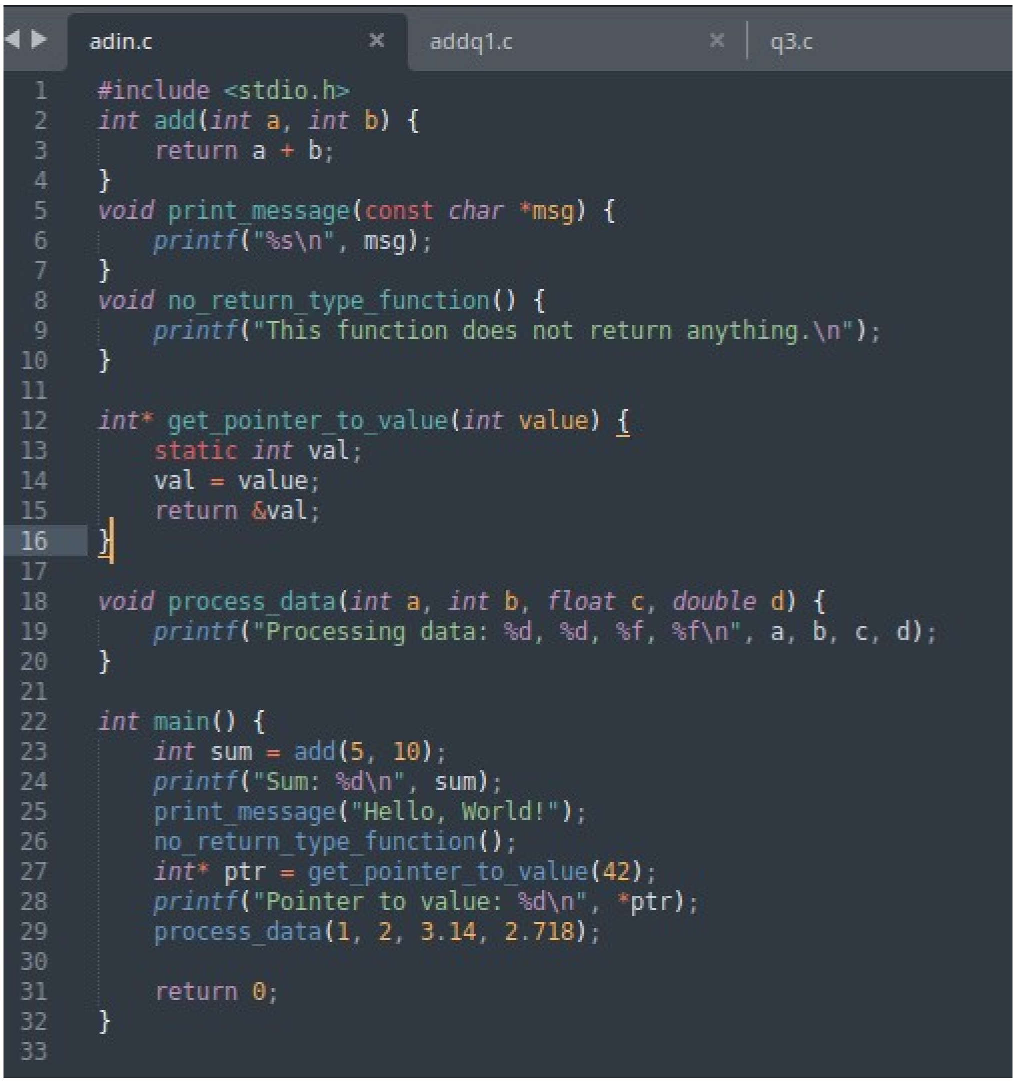


```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAX_LINE_LENGTH 1024
int is_valid_char_for_identifier(char c) {
    return isal
}
void extract_fu
    char return
    char function_name[MAX_LINE_LENGTH];
    char args[MAX_LINE_LENGTH];
    int i = 0, j = 0;
    while (isspace(line[i])) i++;
    while (isalnum(line[i]) || line[i] == '_') {
        return_type[j++] = line[i++];
    }
    return_type[j] = '\0';
    while (isspace(line[i])) i++;
    j = 0;
    while (is_valid_char_for_identifier(line[i])) {
        function_name[j++] = line[i++];
    }
    function_name[j] = '\0';
    while (isspace(line[i])) i++;
    if (line[i] == '(') i++;
    j = 0;
    int paren_count = 1;
    while (line[i] != ')' && line[i] != '\0') {
        if (line[i] == '(') {
            paren_count++;
        } else if (line[i] == ')') {
            paren_count--;
        }
        args[j++] = line[i++];
        if (paren_count == 0) break;
    }
    args[j] = '\0';
    int num_args = 0;
    char *arg = strtok(args, ",");
    while (arg != NULL) {
        num_args++;
        arg = strtok(NULL, ",");
    }
    if (strlen(function_name) > 0) {
        printf("Function Name: %s\n", function_name);
        printf("Return Type: %s\n", return_type);
        printf("Number of Arguments: %d\n\n", num_args);
    }
```



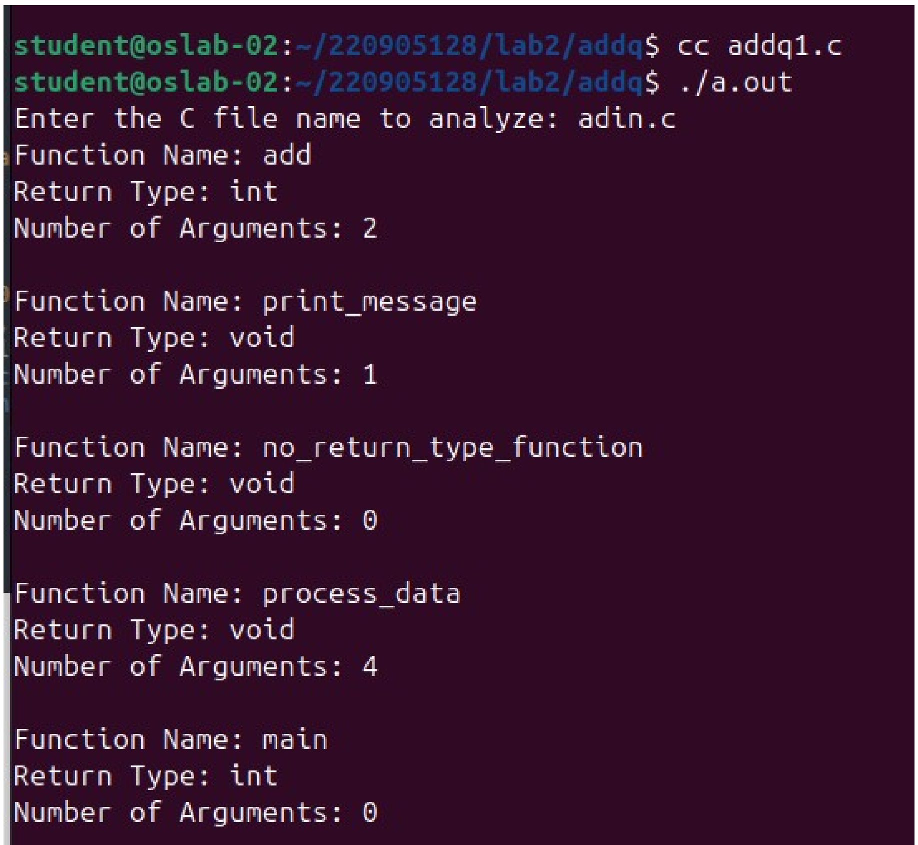
```c
    int paren_count = 1;
    while (line[i] != ')' && line[i] != '\0') {
        if (line[i] == '(') {
            paren_count++;
        } else if (line[i] == ')') {
            paren_count--;
        }
        args[j++] = line[i++];
        if (paren_count == 0) break;
    }
    args[j] = '\0';
    int num_args = 0;
    char *arg = strtok(args, ",");
    while (arg != NULL) {
        num_args++;
        arg = strtok(NULL, ",");
    }
    if (strlen(function_name) > 0) {
        printf("Function Name: %s\n", function_name);
        printf("Return Type: %s\n", return_type);
        printf("Number of Arguments: %d\n\n", num_args);
    }
}
void process_file(FILE *file) {
    char line[MAX_LINE_LENGTH];
    while (fgets(line, sizeof(line), file)) {
        if (line[0] == '\0' || line[0] == '/' || line[0] == '\n') {
            continue;
        }
        if (strchr(line, '(') && strchr(line, ')') && !strchr(line, ';')) {
            extract_function_details(line);
        }
    }
}

int main() {
    char filename[MAX_LINE_LENGTH];
    printf("Enter the C file name to analyze: ");
    scanf("%s", filename);
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("Could not open file %s\n", filename);
        return 1;
    }
    process_file(file);
    fclose(file);
    return 0;
}
```

Input file: adin.c



```c
#include <stdio.h>
int add(int a, int b) {
    return a + b;
}
void print_message(const char *msg) {
    printf("%s\n", msg);
}
void no_return_type_function() {
    printf("This function does not return anything.\n");
}

int* get_pointer_to_value(int value) {
    static int val;
    val = value;
    return &val;
}

void process_data(int a, int b, float c, double d) {
    printf("Processing data: %d, %d, %f, %f\n", a, b, c, d);
}

int main() {
    int sum = add(5, 10);
    printf("Sum: %d\n", sum);
    print_message("Hello, World!");
    no_return_type_function();
    int* ptr = get_pointer_to_value(42);
    printf("Pointer to value: %d\n", *ptr);
    process_data(1, 2, 3.14, 2.718);

    return 0;
}
```

Output:



```
student@oslab-02:~/220905128/lab2/addq$ cc addq1.c
student@oslab-02:~/220905128/lab2/addq$ ./a.out
Enter the C file name to analyze: adin.c
Function Name: add
Return Type: int
Number of Arguments: 2

Function Name: print_message
Return Type: void
Number of Arguments: 1

Function Name: no_return_type_function
Return Type: void
Number of Arguments: 0

Function Name: process_data
Return Type: void
Number of Arguments: 4

Function Name: main
Return Type: int
Number of Arguments: 0
```