Compiler Design Lab

Aditya Agarwal **Roll No. 14** **Reg No. 220905106**

LAB2 - PRELIMINARY SCANNING APPLICATIONS

Q1 - That takes a file as input and replaces blank spaces and tabs by single space and writes the output to a file.

CODE:

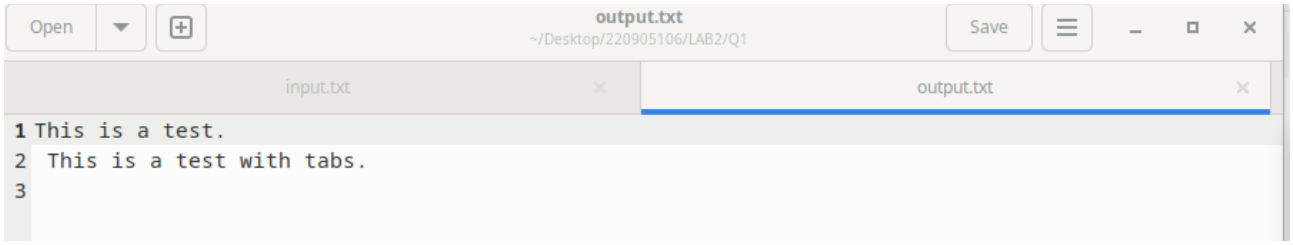
```
#include <stdio.h>
#include <stdlib.h>

int main() {
    FILE *fa, *fb;
    int ca;

    // Open the input file in read mode
    fa = fopen("input.txt", "r");
    if (fa == NULL) {
        printf("Cannot open input file\n");
        exit(0);
    }

    // Open the output file in write mode
    fb = fopen("output.txt", "w");
    if (fb == NULL) {
        printf("Cannot open output file\n");
        exit(0);
    }

    // Read the first character from the input file
    ca = fgetc(fa);

    // Flag to track if the previous character was a space or tab
    int lastWasSpace = 0;

    // Process the file until EOF
    while (ca != EOF) {
        // If the character is a space or tab, replace it with a single space
        if (ca == ' ' || ca == '\t') {
            // Only write a space if the last character was not a space
            if (!lastWasSpace) {
                fputc(' ', fb);
                lastWasSpace = 1;
            }
        } else {
            // Otherwise, write the character to the output file
            fputc(ca, fb);
```

```
            lastWasSpace = 0;
        }

        // Read the next character from the input file
        ca = fgetc(fa);
    }

    // Close the files
    fclose(fa);
    fclose(fb);

    printf("Processing complete. Output written to 'output.txt'.\n");

    return 0;
}
```

INPUT FILE : "INPUT.TXT"

```
This    is    a    test.
   This    is a   test    with   tabs.
```

OUTPUT FILE: "OUTPUT.TXT"



TERMINAL :

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

CODE:

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

int main() {
    FILE *fa, *fb;
    int ca;

    // Open the input file in read mode
    fa = fopen("input.c", "r");
    if (fa == NULL) {
        printf("Cannot open input file\n");
        exit(0);
    }

    // Open the output file in write mode
    fb = fopen("output.c", "w");
    if (fb == NULL) {
        printf("Cannot open output file\n");
        exit(0);
    }

    // Read the first character from the input file
    ca = fgetc(fa);

    // Process the file character by character
    while (ca != EOF) {
        // If the character is '#', it's a preprocessor directive
        if (ca == '#') {
            // Skip the entire preprocessor directive
            while (ca != '\n' && ca != EOF) {
                ca = fgetc(fa);
            }
        } else {
            // Write the character to the output file if it's not part of a directive
            fputc(ca, fb);
        }

        // Read the next character
        ca = fgetc(fa);
    }

    // Close the files
    fclose(fa);
    fclose(fb);

    printf("Preprocessor directives discarded. Output written to 'output.c'.\n");

    return 0; }
```

INPUT FILE : “INPUT.C”

input.c (~/Desktop/220905106/LAB2/Q2) - GVIM1

File Edit Tools Syntax Buffers Window Help

```c
#include <stdio.h>
#define MAX 100

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

OUTPUT FILE: “OUTPUT.C”

output.c (~/Desktop/220905106/LAB2/Q2) - GVIM2

File Edit Tools Syntax Buffers Window Help

```c

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

TERMINAL :

CD_LAB_B1@debianpc-02: ~/Desktop/220905106/LAB2/Q2

File Edit View Search Terminal Help

```
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q2$ bash
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q2$ ls
input.c  output.c  q2  q2.c
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q2$ gcc -o q2 q2.c
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q2$ ./q2
Preprocessor directives discarded. Output written to 'output.c'.
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q2$
```

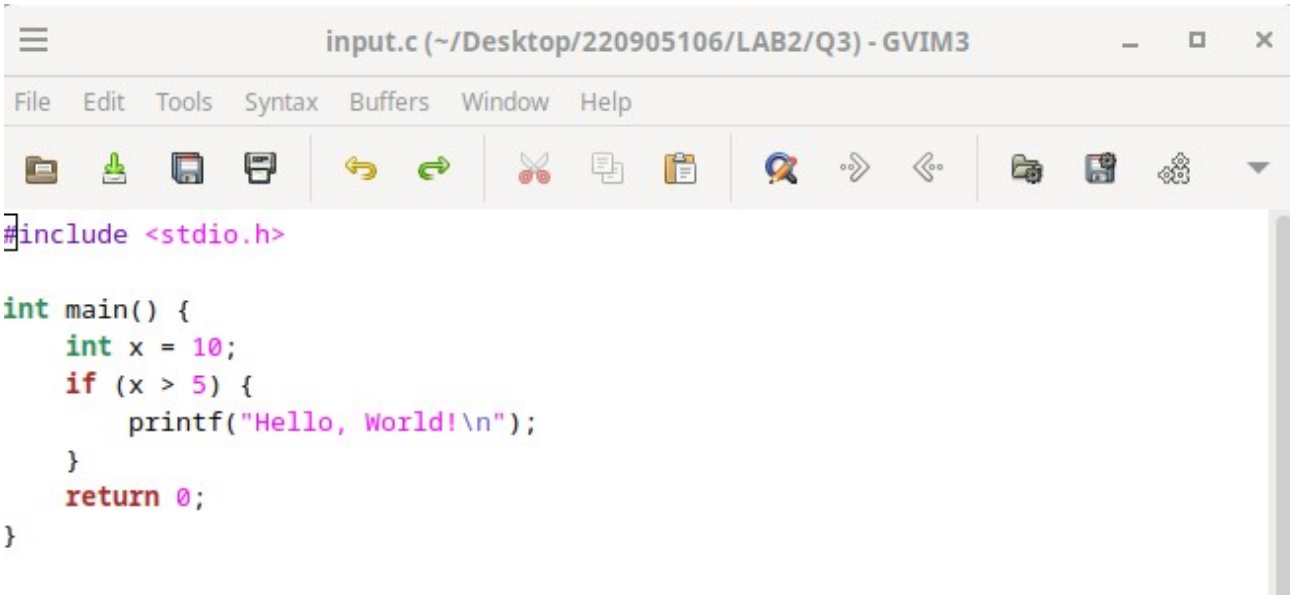
Q3 - That takes C program as input, recognizes all the keywords and prints them in upper case.

CODE:

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

#define MAX_KEYWORDS 32

// List of C keywords
const char *keywords[MAX_KEYWORDS] = {
    "auto", "break", "case", "char", "const", "continue", "default", "do", "double",
    "else", "enum", "extern", "for", "goto", "if", "inline", "int", "long",
    "register", "restrict", "return", "short", "signed", "sizeof", "static",
    "struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while"
};

// Function to check if a word is a C keyword
int is_keyword(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; // Not a keyword
}

// Function to convert a string to uppercase
void to_uppercase(char *str) {
    while (*str) {
        *str = toupper((unsigned char) *str);
        str++;
    }
}

int main() {
    FILE *fa, *fb;
    char word[100]; // Buffer to store words
    int ca, i = 0;

    // Open the input file in read mode
    fa = fopen("input.c", "r");
    if (fa == NULL) {
        printf("Cannot open input file\n");
        exit(0);
    }

    // Open the output file in write mode
    fb = fopen("output.c", "w");
```

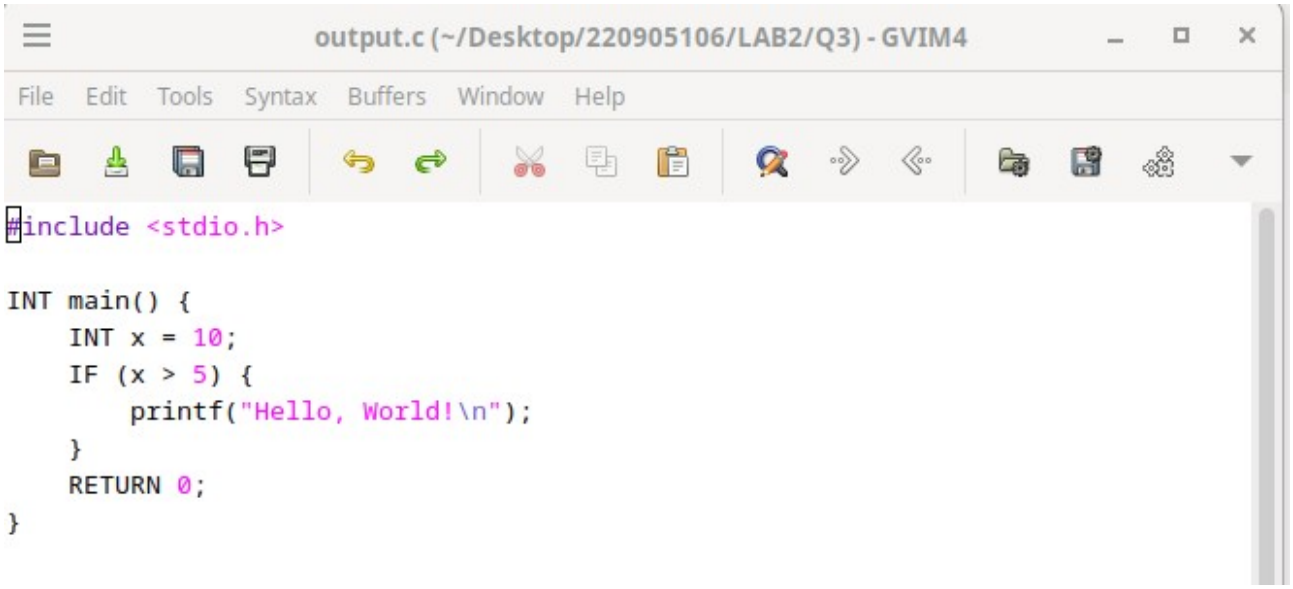
```
    if (fb == NULL) {
        printf("Cannot open output file\n");
        exit(0);
    }

    // Read characters from the input file
    ca = fgetc(fa);
    while (ca != EOF) {
        // If the character is part of a word (alphanumeric or underscore), collect it
        if (isalnum(ca) || ca == '_') {
            word[i++] = ca;  // Add character to the word buffer
        } else {
            // If we have a word and it is a keyword, convert it to uppercase
            if (i > 0) {
                word[i] = '\0'; // Null-terminate the word
                if (is_keyword(word)) {
                    to_uppercase(word); // Convert to uppercase if it's a keyword
                }
                fputs(word, fb); // Write the word to the output file
                i = 0; // Reset the word index
            }

            // Write non-alphanumeric characters (like spaces, operators, etc.) as they are
            fputc(ca, fb);
        }

        // Read the next character
        ca = fgetc(fa);
    }

    // Handle any remaining word after the last character
    if (i > 0) {
        word[i] = '\0'; // Null-terminate the word
        if (is_keyword(word)) {
            to_uppercase(word); // Convert to uppercase if it's a keyword
        }
        fputs(word, fb); // Write the word to the output file
    }

    // Close the files
    fclose(fa);
    fclose(fb);

    printf("Processing complete. Output written to 'output.c'.\n");

    return 0;
}
```

INPUT FILE: “INPUT.C”




```c
#include <stdio.h>

int main() {
    int x = 10;
    if (x > 5) {
        printf("Hello, World!\n");
    }
    return 0;
}
```

OUTPUT FILE: “OUTPUT.C”




```c
#include <stdio.h>

INT main() {
    INT x = 10;
    IF (x > 5) {
        printf("Hello, World!\n");
    }
    RETURN 0;
}
```

TERMINAL:

CD_LAB_B1@debianpc-02: ~/Desktop/220905106/LAB2/Q3

File Edit View Search Terminal Help

```
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q3$ bash
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q3$ ls
input.c  output.c  q3  q3.c
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q3$ gcc -o q3 q3.c
q3.c:13:77: warning: excess elements in array initializer
   13 | , "switch", "typedef", "union", "unsigned", "void", "volatile", "while"
      |                                                                  ^~~~~~~

q3.c:13:77: note: (near initialization for 'keywords')
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q3$ ./q3
Processing complete. Output written to 'output.c'.
CD_LAB_B1@debianpc-02:~/Desktop/220905106/LAB2/Q3$
```