Plagiarism Detector

Mini Project Report

By

Sai Suman Chitturi 1602-18-733-097

Praneeth Kapila 1602-18-733-116

Course: Design and Analysis of Algorithms

Section: B.E. II/IV CSE-B

Semester: IV

Year: II

Department of Computer Science & Engineering

Vasavi College of Engineering

(Autonomous)

(Approved by A.I.C.T.E)

9-5-81, Ibrahimbagh, Hyderabad-31

2019-20

ACKNOWLEDGEMENT

We take this opportunity with pride and enormous gratitude, to express the deeply embedded feeling and gratefulness to our respectable guide **Dr. V. Sireesha,** Department of Computer Science and Engineering, whose guidance was unforgettable and innovative ideas as well as her constructive suggestions have made the presentation of my thesis a grand success.

We are thankful to **Dr. T. Adilakshmi,** Head of Department (CSE), **Vasavi College of Engineering** for their help during our course work.

Finally, at last but not least express our heart full thanks to the management of our college, **Vasavi College of Engineering** for providing the necessary arrangements and support to complete my seminar work successively.

Table of Contents

Abstract

Plagiarism Detector

Plagiarism: The practice of taking someone else's work or ideas and passing them off as one's own.

Plagiarism became a serious issue now-a-days due to the presence of vast resources easily available on the web, which makes developing plagiarism detection tool a useful and challenging task due to scalability issues.

Plagiarism is a serious problem in many institutions and preventing digital plagiarism requires enormous amount of work.

In this project, we concentrate on implementation of an anti-plagiarism algorithm to detect potential plagiarism. We describe the open architecture that could be used for plagiarism detection on different files.

Finally, we present preliminary results on algorithm implementation on 40+ text documents.

We hope our project will help in controlling plagiarism and contribute to the development of a foolproof anti-plagiarism system.

Introduction

Plagiarism Detector

A Mini Project by: Chitturi Sai Suman, Praneeth Kapila

Plagiarism: Plagiarism refers to using someone else's work and ideas without acknowledging them as the source.

Plagiarism is now a serious issue and has become more common due to increased access to data. Hence our project, titled 'Plagiarism Detector', is more relevant today to combat the increasing rates of plagiarism.

We have employed a very unique yet powerful strategy to compare files, that yields results within no time. The application has been engineered so meticulously that the chance of reporting false plagiarism is almost impossible.

Essentially what the application does is, it compares a pair of files in several aspects and if they match beyond a certain threshold percentage, then plagiarism is reported.

Design Strategy:

We have chosen Dynamic Programming as it is the most relevant design strategy for our project. The time complexity for comparing two text files is O(m*n) where m and n are the number of characters present in pair of text files respectively.

Algorithm:

Longest Common Subsequence is the algorithm that plays vital role in our project. We used it for matching statements in C source codes. This gave better results when compared to naive String compare functions in terms of Accuracy of comparison and time complexity.

Modules:

Our project includes general purpose modules like File Handling, Strings and little bit of Math.

Source Code:

```
#include<stdio.h>
#include<time.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<stdbool.h>
#include<ctype.h>
#include<limits.h>
#include<sys/types.h>
#include<errno.h>
#include<sys/wait.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<dirent.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<sys/shm.h>
#include<sys/sem.h>
#include<pthread.h>
#define or ||
#define and &&
#define not !
#define number_of_functions_matching_limit 2
#define number_of_lines_matching_limit 8
#define number_of_scan_matching_limit 2
#define number_of_print_matching_limit 2
#define number_of_for_matching_limit 2
#define number_of_while_matching_limit 2
#define number_of_if_matching_limit 2
#define number_of_else_matching_limit 2
#define number_of_else_if_matching_limit 2
#define number_of_break_matching_limit 2
#define number_of_int_matching_limit 1
#define number_of_float_matching_limit 1
#define number_of_char_matching_limit 1
#define number_of_long_matching_limit 1
#define number_of_double_matching_limit 1
#define number_of_bool_matching_limit 1
#define number_of_include_matching_limit 0
#define number_of_define_matching_limit 2
#define number_of_struct_matching_limit 0
#define number_of_typedef_matching_limit 2
#define number_of_void_functions_matching_limit 1
#define number_of_int_functions_matching_limit 1
#define number_of_float_functions_matching_limit 1
#define number_of_char_functions_matching_limit 1
#define number_of_bool_functions_matching_limit 1
```

```
#define number_of_long_functions_matching_limit 1
#define number_of_double_functions_matching_limit 1
#define function_match_percentage_limit 85
#define cumulative_match_percentage_limit 90
void strlwr(char *string)
{
    for(int i=0;i<strlen(string);i++)
        string[i]=tolower(string[i]);
}
int max(int a, int b)
{
    return (a>b?a:b);
}
int min(int a, int b)
{
    return (a<b?a:b);
}
typedef struct Function
{
    char name[128];
    char return_type[16];
    int number_of_arguments;
    int number_of_int_arguments;
    int number_of_char_arguments;
    int number_of_float_arguments;
    int number_of_bool_arguments;
    int number_of_long_arguments;
    int number_of_double_arguments;
    int number_of_int_variables;
    int number_of_float_variables;
    int number_of_char_variables;
    int number_of_double_variables;
    int number_of_long_variables;
    int number_of_bool_variables;
    int number_of_print_statements;
    int number_of_scan_statements;
    int number_of_for_loops;
    int number_of_while_loops;
    int number_of_if;
    int number_of_else_if;
    int number_of_else;
    int number_of_break;
    int number_of_continue;
    int number_of_statements;
} Function;
typedef struct File_Data
{
    char name_of_file[32];
    Function function[16];
    int number_of_functions;
    int number_of_lines;
```

```
    int number_of_scan;
    int number_of_print;
    int number_of_for;
    int number_of_while;
    int number_of_if;
    int number_of_else;
    int number_of_else_if;
    int number_of_break;
    int number_of_int;
    int number_of_float;
    int number_of_char;
    int number_of_long;
    int number_of_double;
    int number_of_bool;
    int number_of_include;
    int number_of_define;
    int number_of_struct;
    int number_of_typedef;
    int number_of_void_functions;
    int number_of_int_functions;
    int number_of_float_functions;
    int number_of_char_functions;
    int number_of_bool_functions;
    int number_of_long_functions;
    int number_of_double_functions;
    char line[256][192];
}data;
void clrscr()
{
    system("clear");
}
void newline(int n)
{
    while(n--)
        printf("\n");
}
void tab(int n)
{
    while(n--)
        printf("\t");
}
void space(int n)
{
    while(n--)
        printf(" ");
}
void raw_input()
{
    char ch;
    system("/bin/stty raw");
    ch=getchar();
```

```
    system("/bin/stty cooked");
    clrscr();
}
static bool part_of_comment;
bool write_results(data *source_data, data *destination, bool
success,float parameters_matching_percentage,float
function_matching_percentage,float
statement_matching_percentage)
{
    FILE *fptr;
    chdir("..");
    chdir("Results");
    char file_name[96];
    strcpy(file_name,"Result_");
    strcat(file_name,source_data->name_of_file);
    strcat(file_name,"_");
    strcat(file_name,destination->name_of_file);
    strcat(file_name,"_");
    strcat(file_name,".txt");
    fptr=fopen(file_name,"w");
    if(fptr==NULL)
        return false;
    fprintf(fptr,"\n\n\t\t\t");
    if(success)
        fprintf(fptr,"SUCCESS LOG! ");
    else
        fprintf(fptr,"FAILURE LOG! ");
    fprintf(fptr,"Details");
    fprintf(fptr,"\n\n");
    float
cumulative_matching_percentage=parameters_matching_percentage+
function_matching_percentage+statement_matching_percentage;
    cumulative_matching_percentage/=3;
    fprintf(fptr,"\tCumulative Matching Percentage:
%.2f\n",cumulative_matching_percentage);
    fprintf(fptr,"\tParameter Matching Percentage:
%.2f\n",parameters_matching_percentage);
    fprintf(fptr,"\tFunction Matching Percentage:
%.2f\n",function_matching_percentage);
    fprintf(fptr,"\tStatement Matching Percentage:
%.2f\n",statement_matching_percentage);
    fprintf(fptr,"\n");
    fprintf(fptr,"\tDetails of Source File\n\n");
    fprintf(fptr,"Name of File: %s\n",source_data-
>name_of_file);
    fprintf(fptr,"Number of Functions: %d\n",source_data-
>number_of_functions);
    fprintf(fptr,"Function Wise Details\n");
    for(int i=0;i<source_data->number_of_functions;i++)
    {
        fprintf(fptr,"Function %d\n",i+1);
```

```
        fprintf(fptr,"Name of Function: %s\n",source_data-
>function[i].name);
        fprintf(fptr,"Return Type: %s\n",source_data-
>function[i].return_type);
        fprintf(fptr,"Number of Arguments: %d\n",source_data-
>function[i].number_of_arguments);
        fprintf(fptr,"Number of int arguments:
%d\n",source_data->function[i].number_of_int_arguments);
        fprintf(fptr,"Number of char arguments:
%d\n",source_data->function[i].number_of_char_arguments);
        fprintf(fptr,"Number of float Arguments:
%d\n",source_data->function[i].number_of_float_arguments);
        fprintf(fptr,"Number of bool Arguments:
%d\n",source_data->function[i].number_of_bool_arguments);
        fprintf(fptr,"Number of long Arguments:
%d\n",source_data->function[i].number_of_long_arguments);
        fprintf(fptr,"Number of double Arguments:
%d\n",source_data->function[i].number_of_double_arguments);
        fprintf(fptr,"Number of int Variables:
%d\n",source_data->function[i].number_of_int_variables);
        fprintf(fptr,"Number of char Variables:
%d\n",source_data->function[i].number_of_char_variables);
        fprintf(fptr,"Number of float Variables:
%d\n",source_data->function[i].number_of_float_variables);
        fprintf(fptr,"Number of bool Variables:
%d\n",source_data->function[i].number_of_bool_variables);
        fprintf(fptr,"Number of long Variables:
%d\n",source_data->function[i].number_of_long_variables);
        fprintf(fptr,"Number of double Variables:
%d\n",source_data->function[i].number_of_double_variables);
        fprintf(fptr,"Number of printf Statements:
%d\n",source_data->function[i].number_of_print_statements);
        fprintf(fptr,"Number of scanf Statements:
%d\n",source_data->function[i].number_of_scan_statements);
        fprintf(fptr,"Number of for loops: %d\n",source_data-
>function[i].number_of_for_loops);
        fprintf(fptr,"Number of while loops:
%d\n",source_data->function[i].number_of_while_loops);
        fprintf(fptr,"Number of if Statements:
%d\n",source_data->function[i].number_of_if);
        fprintf(fptr,"Number of else if Statements:
%d\n",source_data->function[i].number_of_else_if);
        fprintf(fptr,"Number of else Statements:
%d\n",source_data->function[i].number_of_else);
        fprintf(fptr,"Number of break Statements:
%d\n",source_data->function[i].number_of_break);
        fprintf(fptr,"Number of continue Statements:
%d\n",source_data->function[i].number_of_continue);
        fprintf(fptr,"Total Number of Lines:
%d\n",source_data->function[i].number_of_statements);
        fprintf(fptr,"\n");
```

```
    }
    fprintf(fptr,"\tProgram Details Continued...\n");
    fprintf(fptr,"Number of Lines: %d\n",source_data-
>number_of_lines);
    fprintf(fptr,"Number of scanf Statements:
%d\n",source_data->number_of_scan);
    fprintf(fptr,"Number of printf Statements:
%d\n",source_data->number_of_print);
    fprintf(fptr,"Number of for loops: %d\n",source_data-
>number_of_for);
    fprintf(fptr,"Number of while loops: %d\n",source_data-
>number_of_while);
    fprintf(fptr,"Number of if Statements: %d\n",source_data-
>number_of_if);
    fprintf(fptr,"Number of else Statements:
%d\n",source_data->number_of_else);
    fprintf(fptr,"Number of else if Statements:
%d\n",source_data->number_of_else_if);
    fprintf(fptr,"Number of break Statements:
%d\n",source_data->number_of_break);
    fprintf(fptr,"Number of int Variables: %d\n",source_data-
>number_of_int);
    fprintf(fptr,"Number of float Variables:
%d\n",source_data->number_of_float);
    fprintf(fptr,"Number of char Variables: %d\n",source_data-
>number_of_char);
    fprintf(fptr,"Number of long Variables: %d\n",source_data-
>number_of_long);
    fprintf(fptr,"Number of double Variables:
%d\n",source_data->number_of_double);
    fprintf(fptr,"Number of bool Variables: %d\n",source_data-
>number_of_bool);
    fprintf(fptr,"Number of Included Header Files:
%d\n",source_data->number_of_include);
    fprintf(fptr,"Number of #define pre-processors:
%d\n",source_data->number_of_define);
    fprintf(fptr,"Number of Structures: %d\n",source_data-
>number_of_struct);
    fprintf(fptr,"Number of typedef: %d\n",source_data-
>number_of_typedef);
    fprintf(fptr,"Number of void Functions: %d\n",source_data-
>number_of_void_functions);
    fprintf(fptr,"Number of int Functions: %d\n",source_data-
>number_of_int_functions);
    fprintf(fptr,"Number of float Functions:
%d\n",source_data->number_of_float_functions);
    fprintf(fptr,"Number of char Functions: %d\n",source_data-
>number_of_char_functions);
    fprintf(fptr,"Number of bool Functions: %d\n",source_data-
>number_of_bool_functions);
```

```
    fprintf(fptr,"Number of long Functions: %d\n",source_data-
>number_of_long_functions);
    fprintf(fptr,"Number of double Functions:
%d\n\n\n",source_data->number_of_double_functions);
    fprintf(fptr,"\tDetails of Destination File\n\n");
    fprintf(fptr,"Name of File: %s\n",destination-
>name_of_file);
    fprintf(fptr,"Number of Functions: %d\n",destination-
>number_of_functions);
    fprintf(fptr,"Function Wise Details\n");
    for(int i=0;i<destination->number_of_functions;i++)
    {
        fprintf(fptr,"Function %d\n",i+1);
        fprintf(fptr,"Name of Function: %s\n",destination-
>function[i].name);
        fprintf(fptr,"Return Type: %s\n",destination-
>function[i].return_type);
        fprintf(fptr,"Number of Arguments: %d\n",destination-
>function[i].number_of_arguments);
        fprintf(fptr,"Number of int arguments:
%d\n",destination->function[i].number_of_int_arguments);
        fprintf(fptr,"Number of char arguments:
%d\n",destination->function[i].number_of_char_arguments);
        fprintf(fptr,"Number of float Arguments:
%d\n",destination->function[i].number_of_float_arguments);
        fprintf(fptr,"Number of bool Arguments:
%d\n",destination->function[i].number_of_bool_arguments);
        fprintf(fptr,"Number of long Arguments:
%d\n",destination->function[i].number_of_long_arguments);
        fprintf(fptr,"Number of double Arguments:
%d\n",destination->function[i].number_of_double_arguments);
        fprintf(fptr,"Number of int Variables:
%d\n",destination->function[i].number_of_int_variables);
        fprintf(fptr,"Number of char Variables:
%d\n",destination->function[i].number_of_char_variables);
        fprintf(fptr,"Number of float Variables:
%d\n",destination->function[i].number_of_float_variables);
        fprintf(fptr,"Number of bool Variables:
%d\n",destination->function[i].number_of_bool_variables);
        fprintf(fptr,"Number of long Variables:
%d\n",destination->function[i].number_of_long_variables);
        fprintf(fptr,"Number of double Variables:
%d\n",destination->function[i].number_of_double_variables);
        fprintf(fptr,"Number of printf Statements:
%d\n",destination->function[i].number_of_print_statements);
        fprintf(fptr,"Number of scanf Statements:
%d\n",destination->function[i].number_of_scan_statements);
        fprintf(fptr,"Number of for loops: %d\n",destination-
>function[i].number_of_for_loops);
        fprintf(fptr,"Number of while loops:
%d\n",destination->function[i].number_of_while_loops);
```

```
        fprintf(fptr,"Number of if Statements:
%d\n",destination->function[i].number_of_if);
        fprintf(fptr,"Number of else if Statements:
%d\n",destination->function[i].number_of_else_if);
        fprintf(fptr,"Number of else Statements:
%d\n",destination->function[i].number_of_else);
        fprintf(fptr,"Number of break Statements:
%d\n",destination->function[i].number_of_break);
        fprintf(fptr,"Number of continue Statements:
%d\n",destination->function[i].number_of_continue);
        fprintf(fptr,"Total Number of Lines:
%d\n",destination->function[i].number_of_statements);
        fprintf(fptr,"\n");
    }
    fprintf(fptr,"\tProgram Details Continued...\n");
    fprintf(fptr,"Number of Lines: %d\n",destination-
>number_of_lines);
    fprintf(fptr,"Number of scanf Statements:
%d\n",destination->number_of_scan);
    fprintf(fptr,"Number of printf Statements:
%d\n",destination->number_of_print);
    fprintf(fptr,"Number of for loops: %d\n",destination-
>number_of_for);
    fprintf(fptr,"Number of while loops: %d\n",destination-
>number_of_while);
    fprintf(fptr,"Number of if Statements: %d\n",destination-
>number_of_if);
    fprintf(fptr,"Number of else Statements:
%d\n",destination->number_of_else);
    fprintf(fptr,"Number of else if Statements:
%d\n",destination->number_of_else_if);
    fprintf(fptr,"Number of break Statements:
%d\n",destination->number_of_break);
    fprintf(fptr,"Number of int Variables: %d\n",destination-
>number_of_int);
    fprintf(fptr,"Number of float Variables:
%d\n",destination->number_of_float);
    fprintf(fptr,"Number of char Variables: %d\n",destination-
>number_of_char);
    fprintf(fptr,"Number of long Variables: %d\n",destination-
>number_of_long);
    fprintf(fptr,"Number of double Variables:
%d\n",destination->number_of_double);
    fprintf(fptr,"Number of bool Variables: %d\n",destination-
>number_of_bool);
    fprintf(fptr,"Number of Included Header Files:
%d\n",destination->number_of_include);
    fprintf(fptr,"Number of #define pre-processors:
%d\n",destination->number_of_define);
    fprintf(fptr,"Number of Structures: %d\n",destination-
>number_of_struct);
```

```
    fprintf(fptr,"Number of typedef: %d\n",destination-
>number_of_typedef);
    fprintf(fptr,"Number of void Functions: %d\n",destination-
>number_of_void_functions);
    fprintf(fptr,"Number of int Functions: %d\n",destination-
>number_of_int_functions);
    fprintf(fptr,"Number of float Functions:
%d\n",destination->number_of_float_functions);
    fprintf(fptr,"Number of char Functions: %d\n",destination-
>number_of_char_functions);
    fprintf(fptr,"Number of bool Functions: %d\n",destination-
>number_of_bool_functions);
    fprintf(fptr,"Number of long Functions: %d\n",destination-
>number_of_long_functions);
    fprintf(fptr,"Number of double Functions:
%d\n",destination->number_of_double_functions);
    fprintf(fptr,"\n\n\t\t\t\t*****End of Log*****");
    fclose(fptr);
    chdir("..");
    chdir("Files_to_be_checked");
    return true;
}
int display_files(char *folder_name)
{
    newline(2);
    tab(3);
    int number_of_files=0;
    struct dirent **name_list;
    int n;
    int i=0;
    int j=0;
    n = scandir(folder_name, &name_list,NULL,alphasort);
    printf("Current Directory:\t%s",getcwd(NULL,0));
    newline(2);
    tab(5);
    printf("Files Present in Current Directory");
    newline(3);
    int number_of_columns=4;
    int column_length=30;
    tab(1);
    while(i<n)
    {
        if(strcmp(name_list[i]->d_name,"..")==0 ||
strcmp(name_list[i]->d_name,".")==0)
        {
            free(name_list[i]);
            ++i;
            continue;
        }
        printf("%s", name_list[i]->d_name);
        space(column_length-(strlen(name_list[i]->d_name)));
```

```
        free(name_list[i]);
        ++i;
        ++j;
        if(j%number_of_columns==0)
        {
            newline(2);
            tab(1);
        }
    }
    free(name_list);
    newline(3);
    tab(3);
    return (n-2);
}
void display_introduction()
{
    FILE *fptr;
    fptr=fopen("Introduction.txt","r");
    char ch;
    clrscr();
    while(fscanf(fptr,"%c",&ch)!=EOF)
        printf("%c",ch);
    fclose(fptr);
    newline(2);
    tab(3);
    printf("Press any key to Continue...");
    raw_input();
}
void display_conclusion()
{
    FILE *fptr;
    fptr=fopen("Conclusion.txt","r");
    char ch;
    clrscr();
    while(fscanf(fptr,"%c",&ch)!=EOF)
        printf("%c",ch);
    fclose(fptr);
    newline(2);
    tab(3);
    printf("Press any key to Quit...");
    raw_input();
}
bool prompt_continue()
{
    clrscr();
    newline(3);
    tab(3);
    printf("Would you like to Continue..?\t\t");
    char ch;
    system("/bin/stty raw");
    ch=getchar();
```

```
    system("/bin/stty cooked");
    return (ch=='Y' or ch=='y');
}
int count_number_of_variables(char line[])
{
    int count=0;
    for(int i=0;i<strlen(line);i++)
        if(line[i]==',')
            count+=1;
    return count+1;
}
void get_function_details(Function * function,char
line[256][192], const int pos)
{
    int i=pos;
    int j=0;
    while(line[i][j]!='\0' and line[i][j]!=' ')
        j++;
    j++;
    char name[128];
    int index=0;
    while(line[i][j]!='\0' and line[i][j]!='(')
    {
        name[index]=line[i][j];
        index++;
        j++;
    }
    name[index]='\0';
    int param_pos=j;
    strcpy(function->name,name);
    if(strstr(line[i],"void"))
        strcpy(function->return_type,"void");
    else if(strstr(line[i],"int"))
        strcpy(function->return_type,"int");
    else if(strstr(line[i],"float"))
        strcpy(function->return_type,"float");
    else if(strstr(line[i],"double"))
        strcpy(function->return_type,"double");
    else if(strstr(line[i],"long"))
        strcpy(function->return_type,"long");
    else if(strstr(line[i],"char"))
        strcpy(function->return_type,"char");
    j=0;
    int number_of_arguments=0;
    while(j<strlen(line[i]))
    {
        if(line[i][j]==',')
            number_of_arguments+=1;
        j+=1;
    }
    function->number_of_arguments=(number_of_arguments+1);
```

```
    int count;
    j=param_pos;
    count=0;
    while(j<=strlen(line[i])-strlen("int"))
    {
        if(line[i][j]=='i' and line[i][j+1]=='n' and
line[i][j+2]=='t')
            count+=1;
        j+=1;
    }
    function->number_of_int_arguments=count;
    count=0;
    j=param_pos;
    while(j<=strlen(line[i])-strlen("char"))
    {
        if(line[i][j]=='c' and line[i][j+1]=='h' and
line[i][j+2]=='a' and line[i][j+3]=='r')
            count+=1;
        j+=1;
    }
    function->number_of_char_arguments=count;
    j=param_pos;
    count=0;
    while(j<=strlen(line[i])-strlen("float"))
    {
        if(line[i][j]=='f' and line[i][j+1]=='l' and
line[i][j+2]=='o' and line[i][j+3]=='a' and line[i][j+4]=='t')
            count+=1;
        j+=1;
    }
    function->number_of_float_arguments=count;
    j=param_pos;
    count=0;
    while(j<=strlen(line[i])-strlen("bool"))
    {
        if(line[i][j]=='b' and line[i][j+1]=='o' and
line[i][j+2]=='o' and line[i][j+3]=='l')
            count+=1;
        j+=1;
    }
    function->number_of_bool_arguments=count;
    j=param_pos;
    count=0;
    while(j<=strlen(line[i])-strlen("long"))
    {
        if(line[i][j]=='l' and line[i][j+1]=='o' and
line[i][j+2]=='n' and line[i][j+3]=='g')
            count+=1;
        j+=1;
    }
    function->number_of_long_arguments=count;
```

```
    j=param_pos;
    count=0;
    while(j<=strlen(line[i])-strlen("double"))
    {
        if(line[i][j]=='d' and line[i][j+1]=='o' and
line[i][j+2]=='u' and line[i][j+3]=='b' and line[i][j+4]=='l'
and line[i][j+5]=='e')
            count+=1;
        j+=1;
    }
    function->number_of_double_arguments=count;
    count=0;
    int number_of_statements=0;
    int balanced=0;
    function->number_of_print_statements=0;
    function->number_of_scan_statements=0;
    function->number_of_for_loops=0;
    function->number_of_while_loops=0;
    function->number_of_if=0;
    function->number_of_else_if=0;
    function->number_of_else=0;
    function->number_of_break=0;
    function->number_of_continue=0;
    function->number_of_statements=0;
    i=pos+1;
    do
    {
        if(strstr(line[i],"{"))
            balanced++;
        if(strstr(line[i],"}"))
            balanced--;
        if(strstr(line[i],"printf("))
            function->number_of_print_statements+=1;
        if(strstr(line[i],"scanf("))
            function->number_of_scan_statements+=1;
        if(strstr(line[i],"for("))
            function->number_of_for_loops+=1;
        if(strstr(line[i],"while("))
            function->number_of_while_loops+=1;
        if(strstr(line[i],"if("))
            function->number_of_if+=1;
        if(strstr(line[i],"else if("))
            function->number_of_else_if+=1;
        else if(strstr(line[i],"else"))
            function->number_of_else+=1;
        if(strstr(line[i],"break;"))
            function->number_of_break+=1;
        if(strstr(line[i],"continue;"))
            function->number_of_continue+=1;
        function->number_of_statements+=1;
        i++;
```

```
    }
    while(balanced>0);
    function->number_of_int_variables=0;
    function->number_of_float_variables=0;
    function->number_of_char_variables=0;
    function->number_of_bool_variables=0;
    function->number_of_double_variables=0;
    function->number_of_long_variables=0;
    for(i=pos;i<pos+function->number_of_statements;i++)
    {
        if(strstr(line[i],"int") and strstr(line[i],";"))
            function-
>number_of_int_variables+=count_number_of_variables(line[i]);
        else if(strstr(line[i],"float") and
strstr(line[i],";"))
            function-
>number_of_float_variables+=count_number_of_variables(line[i])
;
        else if(strstr(line[i],"char") and
strstr(line[i],";"))
            function-
>number_of_char_variables+=count_number_of_variables(line[i]);
        else if(strstr(line[i],"double") and
strstr(line[i],";"))
            function-
>number_of_double_variables+=count_number_of_variables(line[i]
);
        else if(strstr(line[i],"long") and
strstr(line[i],";"))
            function-
>number_of_long_variables+=count_number_of_variables(line[i]);
        else if(strstr(line[i],"bool") and
strstr(line[i],";"))
            function-
>number_of_bool_variables+=count_number_of_variables(line[i]);
    }
}
bool not_part_of_comment(char line[])
{
    if(strstr(line,"//"))
        return false;
    else if(strstr(line,"/*"))
    {
        part_of_comment=true;
        if(strstr(line,"*/"))
            part_of_comment=false;
        return false;
    }
    else if(strstr(line,"*/"))
    {
        part_of_comment=false;
```

```
        return false;
    }
    else if(part_of_comment)
        return false;
    return true;
}
data * get_complete_details(char
name_of_file_given_as_input[])
{
    char name_of_file[16];
    strcpy(name_of_file,name_of_file_given_as_input);
    FILE *source=fopen(name_of_file,"r");
    data *source_data=(data *)malloc(sizeof(data));
    int i=0;
    char trash;
    int number_of_lines=0;
    char line[192];
    clrscr();
    part_of_comment=false;
    while(fgets(source_data->line[i],192,source))
    {
        if(not_part_of_comment(source_data->line[i]))
            number_of_lines+=1;
        else
            --i;
        i+=1;
    }
    fclose(source);
    name_of_file[strlen(name_of_file)-4]='\0';
    strcpy(source_data->name_of_file,name_of_file);
    source_data->number_of_lines=number_of_lines;
    source_data->number_of_scan=0;
    source_data->number_of_print=0;
    source_data->number_of_for=0;
    source_data->number_of_while=0;
    source_data->number_of_if=0;
    source_data->number_of_else=0;
    source_data->number_of_else_if=0;
    source_data->number_of_int=0;
    source_data->number_of_float=0;
    source_data->number_of_char=0;
    source_data->number_of_bool=0;
    source_data->number_of_long=0;
    source_data->number_of_double=0;
    source_data->number_of_include=0;
    source_data->number_of_define=0;
    source_data->number_of_struct=0;
    source_data->number_of_typedef=0;
    source_data->number_of_void_functions=0;
    source_data->number_of_int_functions=0;
    source_data->number_of_float_functions=0;
```

```
    source_data->number_of_char_functions=0;
    source_data->number_of_bool_functions=0;
    source_data->number_of_long_functions=0;
    source_data->number_of_double_functions=0;
    source_data->number_of_functions=0;
    i=0;
    int number_of_functions=0;
    while(i<source_data->number_of_lines)
    {
        strcpy(line,source_data->line[i]);
        int length=strlen(line);
        if(line[length-1]==')' or line[length-2]==')' or
line[length-3]==')')
        {
            if(strstr(line,"void")==line)
            {
                source_data->number_of_functions+=1;
                source_data->number_of_void_functions+=1;
                get_function_details(&(source_data-
>function[number_of_functions]),source_data->line,i);
                number_of_functions+=1;
            }
            else if(strstr(line,"int")==line)
            {
                source_data->number_of_functions+=1;
                source_data->number_of_int_functions+=1;
                get_function_details(&(source_data-
>function[number_of_functions]),source_data->line,i);
                number_of_functions+=1;
            }
            else if(strstr(line,"float")==line)
            {
                source_data->number_of_functions+=1;
                source_data->number_of_float_functions+=1;
                get_function_details(&(source_data-
>function[number_of_functions]),source_data->line,i);
                number_of_functions+=1;
            }
            else if(strstr(line,"char")==line)
            {
                source_data->number_of_functions+=1;
                source_data->number_of_char_functions+=1;
                get_function_details(&(source_data-
>function[number_of_functions]),source_data->line,i);
                number_of_functions+=1;
            }
            else if(strstr(line,"bool")==line)
            {
                source_data->number_of_functions+=1;
                source_data->number_of_bool_functions+=1;
```

```
                get_function_details(&(source_data-
>function[number_of_functions]),source_data->line,i);
                number_of_functions+=1;
            }
            else if(strstr(line,"long")==line)
            {
                source_data->number_of_functions+=1;
                source_data->number_of_long_functions+=1;
                get_function_details(&(source_data-
>function[number_of_functions]),source_data->line,i);
                number_of_functions+=1;
            }
            else if(strstr(line,"double")==line)
            {
                source_data->number_of_functions+=1;
                source_data->number_of_double_functions+=1;
                get_function_details(&(source_data-
>function[number_of_functions]),source_data->line,i);
                number_of_functions+=1;
            }
        }
        else if(strstr(line,"scanf("))
            source_data->number_of_scan+=1;
        else if(strstr(line,"printf("))
            source_data->number_of_print+=1;
        else if(strstr(line,"for("))
            source_data->number_of_for+=1;
        else if(strstr(line,"while("))
            source_data->number_of_while+=1;
        else if(strstr(line,"if("))
            source_data->number_of_if+=1;
        else if(strstr(line,"else if("))
            source_data->number_of_else_if+=1;
        else if(strstr(line,"else"))
            source_data->number_of_else+=1;
        else if(strstr(line,"break;"))
            source_data->number_of_break+=1;
        else if(strstr(line,"#include"))
            source_data->number_of_include+=1;
        else if(strstr(line,"#define"))
            source_data->number_of_define+=1;
        else if(strstr(line,"struct"))
            source_data->number_of_struct+=1;
        else if(strstr(line,"typedef"))
            source_data->number_of_typedef+=1;
        else if(strstr(line,"int") and strstr(line,";"))
            source_data-
>number_of_int+=count_number_of_variables(line);
        else if(strstr(line,"float") and strstr(line,";"))
            source_data-
>number_of_float+=count_number_of_variables(line);
```

```
        else if(strstr(line,"char") and strstr(line,";"))
            source_data-
>number_of_char+=count_number_of_variables(line);
        else if(strstr(line,"bool") and strstr(line,";"))
            source_data-
>number_of_bool+=count_number_of_variables(line);
        else if(strstr(line,"long") and strstr(line,";"))
            source_data-
>number_of_long+=count_number_of_variables(line);
        else if(strstr(line,"double") and strstr(line,";"))
            source_data-
>number_of_double+=count_number_of_variables(line);
        i+=1;
    }
    return source_data;
}
int function_compare(Function *function1, Function *function2)
{
    int number_of_matching=0;
    strlwr(function1->name);
    strlwr(function2->name);
    if(strcmp(function1->name,function2->name)==0)
        number_of_matching+=1;
    if(strcmp(function1->return_type,function2-
>return_type)==0)
        number_of_matching+=1;
    if(function1->number_of_arguments==function2-
>number_of_arguments)
        number_of_matching+=1;
    if(function1->number_of_int_arguments==function2-
>number_of_int_arguments)
        number_of_matching+=1;
    if(function1->number_of_char_arguments==function2-
>number_of_char_arguments)
        number_of_matching+=1;
    if(function1->number_of_float_arguments==function2-
>number_of_float_arguments)
        number_of_matching+=1;
    if(function1->number_of_bool_arguments==function2-
>number_of_bool_arguments)
        number_of_matching+=1;
    if(function1->number_of_long_arguments==function2-
>number_of_long_arguments)
        number_of_matching+=1;
    if(function1->number_of_double_arguments==function2-
>number_of_double_arguments)
        number_of_matching+=1;
    if(function1->number_of_int_variables==function2-
>number_of_int_variables)
        number_of_matching+=1;
```

```
    if(function1->number_of_float_variables==function2-
>number_of_float_variables)
        number_of_matching+=1;
    if(function1->number_of_char_variables==function2-
>number_of_char_variables)
        number_of_matching+=1;
    if(function1->number_of_double_variables==function2-
>number_of_double_variables)
        number_of_matching+=1;
    if(function1->number_of_long_variables==function2-
>number_of_long_variables)
        number_of_matching+=1;
    if(function1->number_of_bool_variables==function2-
>number_of_bool_variables)
        number_of_matching+=1;
    if(function1->number_of_print_statements==function2-
>number_of_print_statements)
        number_of_matching+=1;
    if(function1->number_of_scan_statements==function2-
>number_of_scan_statements)
        number_of_matching+=1;
    if(function1->number_of_for_loops==function2-
>number_of_for_loops)
        number_of_matching+=1;
    if(function1->number_of_while_loops==function2-
>number_of_while_loops)
        number_of_matching+=1;
    if(function1->number_of_if==function2->number_of_if)
        number_of_matching+=1;
    if(function1->number_of_else_if==function2-
>number_of_else_if)
        number_of_matching+=1;
    if(function1->number_of_else==function2->number_of_else)
        number_of_matching+=1;
    if(function1->number_of_break==function2->number_of_break)
        number_of_matching+=1;
    if(function1->number_of_continue==function2-
>number_of_continue)
        number_of_matching+=1;
    if(function1->number_of_statements==function2-
>number_of_statements)
        number_of_matching+=1;
    return number_of_matching;
}
int longest_common_subseqence( char *string1, char *string2,
int m, int n)
{
    int l[m+1][n+1];
    int i,j;
    for(i=0;i<=m;i++)
    {
```

```
        for(j=0;j<=n;j++)
        {
            if(i==0 or j==0)
                l[i][j]=0;
            else if(string1[i-1]==string2[j-1])
                l[i][j]=l[i-1][j-1]+1;
            else
                l[i][j]=max(l[i-1][j],l[i][j-1]);
        }
    }
    return l[m][n];
}
float statement_compare(data *source_data, data *destination)
{
    float statement_matching_percentage;
    int number_of_statements_matching=0;
    int source_line,destination_line;
    int maximum_lines=max(source_data-
>number_of_lines,destination->number_of_lines);
    if(maximum_lines==0)
        return 0;
    for(source_line=0;source_line<source_data-
>number_of_lines;source_line++)
    {
        for(destination_line=0;destination_line<destination-
>number_of_lines;destination_line++)
        {
            int length1=strlen(source_data-
>line[source_line]);
            int length2=strlen(destination-
>line[destination_line]);
            int maximum=max(length1,length2);
            if(longest_common_subseqence(source_data-
>line[source_line],destination-
>line[destination_line],length1,length2)>=(maximum-
(0.1*maximum)))
            {
                number_of_statements_matching+=1;
                break;
            }
        }
    }

statement_matching_percentage=(100*number_of_statements_matchi
ng)/maximum_lines;
    return statement_matching_percentage;
}
void deep_compare(data *source_data, data *destination)
{
    clrscr();
    newline(3);
```

```
    tab(3);
    printf("Comparing %s and %s.....",source_data-
>name_of_file,destination->name_of_file);
    int total_parameters=27;
    int number_of_matching_parameters=0;
    if(abs(source_data->number_of_functions-destination-
>number_of_functions)<=number_of_functions_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_lines-destination-
>number_of_lines)<=number_of_lines_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_scan-destination-
>number_of_scan)<=number_of_scan_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_print-destination-
>number_of_print)<=number_of_print_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_for-destination-
>number_of_for)<=number_of_for_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_while-destination-
>number_of_while)<=number_of_while_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_if-destination-
>number_of_if)<=number_of_if_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_else-destination-
>number_of_else)<=number_of_else_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_else_if-destination-
>number_of_else_if)<=number_of_else_if_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_break-destination-
>number_of_break)<=number_of_break_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_int-destination-
>number_of_int)<=number_of_int_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_float-destination-
>number_of_float)<=number_of_float_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_char-destination-
>number_of_char)<=number_of_char_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_long-destination-
>number_of_long)<=number_of_long_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_double-destination-
>number_of_double)<=number_of_double_matching_limit)
        number_of_matching_parameters+=1;
```

```
    if(abs(source_data->number_of_bool-destination-
>number_of_bool)<=number_of_bool_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_include-destination-
>number_of_include)<=number_of_include_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_define-destination-
>number_of_define)<=number_of_define_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_struct-destination-
>number_of_struct)<=number_of_struct_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_typedef-destination-
>number_of_typedef)<=number_of_typedef_matching_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_void_functions-destination-
>number_of_void_functions)<=number_of_void_functions_matching_
limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_int_functions-destination-
>number_of_int_functions)<=number_of_int_functions_matching_li
mit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_float_functions-destination-
>number_of_float_functions)<=number_of_float_functions_matchin
g_limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_char_functions-destination-
>number_of_char_functions)<=number_of_char_functions_matching_
limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_bool_functions-destination-
>number_of_bool_functions)<=number_of_bool_functions_matching_
limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_long_functions-destination-
>number_of_long_functions)<=number_of_long_functions_matching_
limit)
        number_of_matching_parameters+=1;
    if(abs(source_data->number_of_double_functions-
destination-
>number_of_double_functions)<=number_of_double_functions_match
ing_limit)
        number_of_matching_parameters+=1;
    float parameters_matching_percentage =
((float)((float)(100*number_of_matching_parameters))/((float)(
total_parameters)));
    int total_matching_functions=0;
    int total_function_parameters=25;
    int max_of_functions=max(source_data-
>number_of_functions,destination->number_of_functions);
```

```
    for(int i=0;i<source_data->number_of_functions;i++)
    {
        for(int j=0;j<destination->number_of_functions;j++)
        {

            int
matching_function_parameters=function_compare(&(source_data-
>function[i]),&(destination->function[j]));
            float
current_matching_percentage=(float)(100*matching_function_para
meters/total_function_parameters);

if(current_matching_percentage>function_match_percentage_limit
)
            {
                total_matching_functions+=1;
                break;
            }
        }
    }
    float function_matching_percentage=0;
    float statement_matching_percentage=0;
    float cumulative_matching_percentage=0;
    if(max_of_functions==0);
    else
    {

function_matching_percentage=(float)((total_matching_functions
*100)/max_of_functions);

statement_matching_percentage=statement_compare(source_data,de
stination);

cumulative_matching_percentage=parameters_matching_percentage+
function_matching_percentage+statement_matching_percentage;
        cumulative_matching_percentage/=3;
    }
    newline(3);
    tab(4);

if(cumulative_matching_percentage>cumulative_match_percentage_
limit)
    {
        printf("PLAGIARISM ACTIVITY CONFIRMED!!!");
        newline(2);
        tab(3);
        printf("Would you like to Write Results to a
File?(Y/N)\t");
        char choice;
        system("/bin/stty raw");
        choice=getchar();
```

```
        system("/bin/stty cooked");
        if(choice=='Y' or choice=='y')
        {
            printf("Writing the Result...");
            newline(2);
            tab(3);

if(write_results(source_data,destination,true,parameters_match
ing_percentage,function_matching_percentage,statement_matching
_percentage))
                printf("Results written Successfully!!!");
            else
                printf("Failure in Writing Results!!!");
        }
    }
    else
    {
        printf("SIMILARITIES AMONG DOCUMENTS IS LESS THAN
SPECIFIED THRESHOLD");
        newline(2);
        tab(3);
        printf("Would you like to Write Results to a
File?(Y/N)\t");
        char choice;
        system("/bin/stty raw");
        choice=getchar();
        system("/bin/stty cooked");
        if(choice=='Y' or choice=='y')
        {
            newline(2);
            tab(3);

if(write_results(source_data,destination,false,parameters_matc
hing_percentage,function_matching_percentage,statement_matchin
g_percentage))
                printf("Results Written Successfully!!!");
            else
                printf("Failure in Writing Results!!!");
        }
    }
    newline(2);
    tab(3);
    printf("Press any key to Continue...");
    raw_input();
}
void report_plagiarism(char name_of_source[], int
number_of_files)
{
    data *source = get_complete_details(name_of_source);
    char name_of_destination[]="file01.txt";
    for(int i=1;i<=number_of_files;i++)
```

```
    {
        if(strcmp(name_of_source,name_of_destination)!=0)
        {
            data *destination =
get_complete_details(name_of_destination);
            deep_compare(source,destination);
            free(destination);
        }
        if(name_of_destination[5]=='9')
        {
            name_of_destination[5]='0';
            name_of_destination[4]+=1;
        }
        else
        {
            name_of_destination[5]+=1;
        }
    }
}
int main()
{
    clrscr();
    display_introduction();
    bool flag=true;
    int number_of_files;
    chdir("Files_to_be_checked");
    char name_of_file[100];
    while(flag)
    {
        clrscr();
        newline(3);
        tab(4);
        printf("This Application detects Plagiarism among Set
of Programming Codes");
        newline(2);
        tab(4);
        printf("Choose a file to search for Plagiarised
Content among others");
        number_of_files=display_files(".");
        printf("Enter the name of the File to be checked:\t");
        scanf("%s",name_of_file);
        char trash;
        scanf("%c",&trash);
        report_plagiarism(name_of_file,number_of_files);
        flag=prompt_continue();
    }
    chdir("..");
    display_conclusion();
    return 0;
}
```

Screenshots

```
                                    PLAGIARISM DETECTOR

                        Project by Chitturi Sai Suman and Praneeth Kapila

PLAGIARISM:

        Plagiarism is defined as the practice of directly copying and then presenting an existing
production without accurate citing or referencing, and/or passing off the product as one's own,
without permission from the original producer.

        Plagiarism is the most frequent offence under the Academic Code of Conduct, as a result of
a lack of proper acknowledgement. By understanding the plagiarism meaning and being able to identify
the plagiarism definition, you can be confident that you will avoid the consequences.

INSPIRATION:

        Plagiarism became a serious issue now a days due to the presence of vast resources easily
available on the web, which makes developing plagiarism detection tool a useful and challengnig
task due to the scalability issues.

        Our project is implementing a Plagiarism Detection Engine oriented for Programming Source codes.
A set of Programming source codes are tested for Plagiarism check and hence reported whether or not the
the pair of codes are Plagiarised.

        Press any key to Continue...
```

1. Introduction

2. Files present in Current Working Directory

```
                This Application detects Plagiarism among Set of Programming Codes

                Choose a file to search for Plagiarised Content among others

        Current Directory:      /home/suman/DAA_Mini_Project/Plagiarism Detector/Files_to_be_checked

                        Files Present in Current Directory

file01.txt              file02.txt              file03.txt              file04.txt

file05.txt              file06.txt              file07.txt              file08.txt

file09.txt              file10.txt              file11.txt              file12.txt

file13.txt              file14.txt              file15.txt              file16.txt

file17.txt              file18.txt              file19.txt              file20.txt

file21.txt              file22.txt              file23.txt              file24.txt


        Enter the name of the File to be checked:
```

```
This Application detects Plagiarism among Set of Programming Codes

Choose a file to search for Plagiarised Content among others

Current Directory:      /home/suman/DAA_Mini_Project/Plagiarism Detector/Files_to_be_checked

Files Present in Current Directory

file01.txt      file02.txt      file03.txt      file04.txt
file05.txt      file06.txt      file07.txt      file08.txt
file09.txt      file10.txt      file11.txt      file12.txt
file13.txt      file14.txt      file15.txt      file16.txt
file17.txt      file18.txt      file19.txt      file20.txt
file21.txt      file22.txt      file23.txt      file24.txt

Enter the name of the File to be checked:      file22.txt
```

3. User selecting file22.txt as the target file

4. Process of detection starts

```
Comparing file22 and file01.....

SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N)
```

```
Comparing file22 and file01.....

        SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N) n

Press any key to Continue...
```

5. Similarities among files is less than required for Plagiarism activity

```
Comparing file22 and file02.....

        SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N) n

Press any key to Continue...
```

```
Comparing file22 and file03.....

        SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N) n

Press any key to Continue...
```

6. Similarities among files is less than required for Plagiarism activity

```
Comparing file22 and file04.....

        SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N) n

Press any key to Continue...
```

This Process of comparison continues until the contents of selected file don't match with that of other files

```
Comparing file22 and file24.....

        PLAGIARISM ACTIVITY CONFIRMED!!!

Would you like to Write Results to a File?(Y/N)
```

7. Plagiarism detected among file22 and file24

```
Comparing file22 and file24.....

        PLAGIARISM ACTIVITY CONFIRMED!!!

Would you like to Write Results to a File?(Y/N) yWriting the Result...

Results written Successfully!!!

Press any key to Continue...
```

8. User opting to write results into a text file. The text file contains the details of both the files

A file named Result_file22_file24.txt is created in the Results Folder. Contents:

```
                    SUCCESS LOG! Details

      Cumulative Matching Percentage: 96.10
      Parameter Matching Percentage: 96.30
      Function Matching Percentage: 100.00
      Statement Matching Percentage: 92.00

      Details of Source File

Name of File: file22
Number of Functions: 2
Function Wise Details
Function 1
Name of Function: long int compare
Return Type: void
Number of Arguments: 2
Number of int arguments: 0
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 2
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of printf Statements: 0
Number of scanf Statements: 0
Number of for loops: 0
Number of while loops: 0
Number of if Statements: 2
Number of else if Statements: 1
Number of else Statements: 0
Number of break Statements: 0
Number of continue Statements: 0
Total Number of Lines: 9

Function 2
Name of Function: main
Return Type: int
Number of Arguments: 1
Number of int arguments: 0
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 12
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of printf Statements: 1
```

```
Number of scanf Statements: 3
Number of for loops: 2
Number of while loops: 1
Number of if Statements: 1
Number of else if Statements: 0
Number of else Statements: 1
Number of break Statements: 0
Number of continue Statements: 1
Total Number of Lines: 29

      Program Details Continued...
Number of Lines: 42
Number of scanf Statements: 1
Number of printf Statements: 0
Number of for loops: 0
Number of while loops: 0
Number of if Statements: 0
Number of else Statements: 1
Number of else if Statements: 0
Number of break Statements: 0
Number of int Variables: 8
Number of float Variables: 0
Number of char Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of bool Variables: 0
Number of Included Header Files: 2
Number of #define pre-processors: 0
Number of Structures: 0
Number of typedef: 0
Number of void Functions: 0
Number of int Functions: 1
Number of float Functions: 0
Number of char Functions: 0
Number of bool Functions: 0
Number of long Functions: 1
Number of double Functions: 0


      Details of Destination File

Name of File: file24
Number of Functions: 2
Function Wise Details
Function 1
Name of Function: long int compare
Return Type: void
Number of Arguments: 2
Number of int arguments: 0
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 2
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of printf Statements: 0
```

```
Number of scanf Statements: 0
Number of for loops: 0
Number of while loops: 0
Number of if Statements: 2
Number of else if Statements: 1
Number of else Statements: 0
Number of break Statements: 0
Number of continue Statements: 0
Total Number of Lines: 9

Function 2
Name of Function: main
Return Type: int
Number of Arguments: 1
Number of int arguments: 0
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 12
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of printf Statements: 1
Number of scanf Statements: 3
Number of for loops: 2
Number of while loops: 1
Number of if Statements: 1
Number of else if Statements: 0
Number of else Statements: 1
Number of break Statements: 0
Number of continue Statements: 1
Total Number of Lines: 29

      Program Details Continued...
Number of Lines: 42
Number of scanf Statements: 1
Number of printf Statements: 0
Number of for loops: 0
Number of while loops: 0
Number of if Statements: 0
Number of else Statements: 1
Number of else if Statements: 0
Number of break Statements: 14
Number of int Variables: 8
Number of float Variables: 0
Number of char Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of bool Variables: 0
Number of Included Header Files: 2
Number of #define pre-processors: 0
Number of Structures: 0
Number of typedef: 0
Number of void Functions: 0
Number of int Functions: 1
Number of float Functions: 0
Number of char Functions: 0
```

```
Number of bool Functions: 0
Number of long Functions: 1
Number of double Functions: 0

                    *****End of Log*****
```

```
Would you like to Continue..?
```

9. Prompting user for running the plagiarism detector over another file

```
This Application detects Plagiarism among Set of Programming Codes
Choose a file to search for Plagiarised Content among others
Current Directory:      /home/suman/DAA_Mini_Project/Plagiarism Detector/Files_to_be_checked
Files Present in Current Directory

file01.txt      file02.txt      file03.txt      file04.txt
file05.txt      file06.txt      file07.txt      file08.txt
file09.txt      file10.txt      file11.txt      file12.txt
file13.txt      file14.txt      file15.txt      file16.txt
file17.txt      file18.txt      file19.txt      file20.txt
file21.txt      file22.txt      file23.txt      file24.txt

Enter the name of the File to be checked:
```

```
This Application detects Plagiarism among Set of Programming Codes

Choose a file to search for Plagiarised Content among others

Current Directory:      /home/suman/DAA_Mini_Project/Plagiarism Detector/Files_to_be_checked

Files Present in Current Directory

file01.txt          file02.txt          file03.txt          file04.txt
file05.txt          file06.txt          file07.txt          file08.txt
file09.txt          file10.txt          file11.txt          file12.txt
file13.txt          file14.txt          file15.txt          file16.txt
file17.txt          file18.txt          file19.txt          file20.txt
file21.txt          file22.txt          file23.txt          file24.txt

Enter the name of the File to be checked:      file06.txt
```

10. User selecting file06.txt as a Source file.

```
Comparing file06 and file01.....

SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N) n

Press any key to Continue...
```

11. The same process of comparison continues.

```
Comparing file06 and file04.....

        SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N)

Press any key to Continue...
```

12. Similarities among the files is not sufficient for Plagiarism

```
Comparing file06 and file05.....

        SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N) n

Press any key to Continue...
```

```
Comparing file06 and file07.....

        SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N) n

Press any key to Continue...
```

13. The Process of Comparison continues until the text files don't match.

```
Comparing file06 and file08.....

        SIMILARITIES AMONG DOCUMENTS IS LESS THAN SPECIFIED THRESHOLD

Would you like to Write Results to a File?(Y/N) n

Press any key to Continue...
```

```
Comparing file06 and file10.....

        PLAGIARISM ACTIVITY CONFIRMED!!!

Would you like to Write Results to a File?(Y/N)
```

14. Plagiarism detected among file06 and file10

```
Comparing file06 and file10.....

        PLAGIARISM ACTIVITY CONFIRMED!!!

Would you like to Write Results to a File?(Y/N) yWriting the Result...

Results written Successfully!!!

Press any key to Continue...
```

15. User opting to write results into a text file. The text file contains the details of both the files

A file named Result_file06_file10.txt is created in the Results Folder. Contents:

```
                    SUCCESS LOG! Details

      Cumulative Matching Percentage: 94.43
      Parameter Matching Percentage: 96.30
      Function Matching Percentage: 100.00
      Statement Matching Percentage: 87.00

      Details of Source File

Name of File: file06
Number of Functions: 2
Function Wise Details
Function 1
Name of Function: countsetbits
Return Type: int
Number of Arguments: 1
Number of int arguments: 1
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 1
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of printf Statements: 0
Number of scanf Statements: 0
Number of for loops: 0
Number of while loops: 1
Number of if Statements: 0
Number of else if Statements: 0
Number of else Statements: 0
Number of break Statements: 0
Number of continue Statements: 0
Total Number of Lines: 9

Function 2
Name of Function:  main
Return Type: int
Number of Arguments: 1
Number of int arguments: 0
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 18
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of printf Statements: 2
```

```
Number of scanf Statements: 4
Number of for loops: 2
Number of while loops: 1
Number of if Statements: 1
Number of else if Statements: 0
Number of else Statements: 1
Number of break Statements: 0
Number of continue Statements: 0
Total Number of Lines: 25

      Program Details Continued...
Number of Lines: 39
Number of scanf Statements: 0
Number of printf Statements: 0
Number of for loops: 0
Number of while loops: 0
Number of if Statements: 0
Number of else Statements: 1
Number of else if Statements: 0
Number of break Statements: 1685353276
Number of int Variables: 13
Number of float Variables: 0
Number of char Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of bool Variables: 0
Number of Included Header Files: 3
Number of #define pre-processors: 0
Number of Structures: 0
Number of typedef: 0
Number of void Functions: 0
Number of int Functions: 2
Number of float Functions: 0
Number of char Functions: 0
Number of bool Functions: 0
Number of long Functions: 0
Number of double Functions: 0


      Details of Destination File

Name of File: file10
Number of Functions: 2
Function Wise Details
Function 1
Name of Function: countsetbits
Return Type: int
Number of Arguments: 1
Number of int arguments: 1
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 1
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of printf Statements: 0
```

```
Number of scanf Statements: 0
Number of for loops: 0
Number of while loops: 1
Number of if Statements: 0
Number of else if Statements: 0
Number of else Statements: 0
Number of break Statements: 0
Number of continue Statements: 0
Total Number of Lines: 9

Function 2
Name of Function:  main
Return Type: int
Number of Arguments: 1
Number of int arguments: 0
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 17
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of printf Statements: 2
Number of scanf Statements: 4
Number of for loops: 2
Number of while loops: 1
Number of if Statements: 1
Number of else if Statements: 0
Number of else Statements: 1
Number of break Statements: 0
Number of continue Statements: 0
Total Number of Lines: 25

      Program Details Continued...
Number of Lines: 40
Number of scanf Statements: 0
Number of printf Statements: 0
Number of for loops: 0
Number of while loops: 0
Number of if Statements: 0
Number of else Statements: 1
Number of else if Statements: 0
Number of break Statements: 538976293
Number of int Variables: 12
Number of float Variables: 0
Number of char Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of bool Variables: 0
Number of Included Header Files: 3
Number of #define pre-processors: 0
Number of Structures: 0
Number of typedef: 0
Number of void Functions: 0
Number of int Functions: 2
Number of float Functions: 0
Number of char Functions: 0
```

```
Number of bool Functions: 0
Number of long Functions: 0
Number of double Functions: 0

                    *****End of Log*****
```

Results file when two files don't match:

Example:

```
                    FAILURE LOG! Details

      Cumulative Matching Percentage: 43.20
      Parameter Matching Percentage: 92.59
      Function Matching Percentage: 0.00
      Statement Matching Percentage: 37.00

      Details of Source File

Name of File: file10
Number of Functions: 2
Function Wise Details
Function 1
Name of Function: countsetbits
Return Type: int
Number of Arguments: 1
Number of int arguments: 1
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 1
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of printf Statements: 0
Number of scanf Statements: 0
Number of for loops: 0
Number of while loops: 1
Number of if Statements: 0
Number of else if Statements: 0
Number of else Statements: 0
Number of break Statements: 0
Number of continue Statements: 0
Total Number of Lines: 9

Function 2
Name of Function:  main
Return Type: int
Number of Arguments: 1
Number of int arguments: 0
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 17
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
```

```
Number of double Variables: 0
Number of printf Statements: 2
Number of scanf Statements: 4
Number of for loops: 2
Number of while loops: 1
Number of if Statements: 1
Number of else if Statements: 0
Number of else Statements: 1
Number of break Statements: 0
Number of continue Statements: 0
Total Number of Lines: 25

      Program Details Continued...
Number of Lines: 40
Number of scanf Statements: 0
Number of printf Statements: 0
Number of for loops: 0
Number of while loops: 0
Number of if Statements: 0
Number of else Statements: 1
Number of else if Statements: 0
Number of break Statements: 0
Number of int Variables: 12
Number of float Variables: 0
Number of char Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of bool Variables: 0
Number of Included Header Files: 3
Number of #define pre-processors: 0
Number of Structures: 0
Number of typedef: 0
Number of void Functions: 0
Number of int Functions: 2
Number of float Functions: 0
Number of char Functions: 0
Number of bool Functions: 0
Number of long Functions: 0
Number of double Functions: 0


      Details of Destination File

Name of File: file01
Number of Functions: 1
Function Wise Details
Function 1
Name of Function: main
Return Type: int
Number of Arguments: 1
Number of int arguments: 0
Number of char arguments: 0
Number of float Arguments: 0
Number of bool Arguments: 0
Number of long Arguments: 0
Number of double Arguments: 0
Number of int Variables: 11
Number of char Variables: 0
Number of float Variables: 0
Number of bool Variables: 0
Number of long Variables: 0
```

```
Number of double Variables: 0
Number of printf Statements: 1
Number of scanf Statements: 3
Number of for loops: 1
Number of while loops: 2
Number of if Statements: 3
Number of else if Statements: 0
Number of else Statements: 0
Number of break Statements: 1
Number of continue Statements: 0
Total Number of Lines: 38

      Program Details Continued...
Number of Lines: 40
Number of scanf Statements: 0
Number of printf Statements: 0
Number of for loops: 0
Number of while loops: 0
Number of if Statements: 0
Number of else Statements: 0
Number of else if Statements: 0
Number of break Statements: 1
Number of int Variables: 9
Number of float Variables: 0
Number of char Variables: 0
Number of long Variables: 0
Number of double Variables: 0
Number of bool Variables: 0
Number of Included Header Files: 1
Number of #define pre-processors: 0
Number of Structures: 0
Number of typedef: 0
Number of void Functions: 0
Number of int Functions: 1
Number of float Functions: 0
Number of char Functions: 0
Number of bool Functions: 0
Number of long Functions: 0
Number of double Functions: 0

                    *****End of Log*****
```

```
Would you like to Continue..?
```

16. Prompting user whether to continue using the application or not.

```
                    Plagiarism Detection

        A Mini-Project by Chitturi Sai Suman, Praneeth Kapila

        We are very grateful for using the Application

    For any Queries or Feedback, please write to saisumanchitturi@gmail.com

Press any key to Quit...
```

17. User selecting to quit, Preview of Conclusion.

Test Cases

Test Case 0: Consider two C Source files.

File1:

```
#include<stdio.h>
int main()
{
    int i,j,t,n;
    long long int answer;
    long long int arr[100000];
    long long int count;
    scanf("%d",&t);
    while(t--)
    {
        count=0;
        int answer=1,r=0;
        scanf("%lld",&n);
        for(i=0;i<n;i++)
        {
            scanf("%lld",&arr[i]);
        }
        i=0;j=0;
        while(i<=j && j<n)
        {
            answer*=arr[j];
            if(answer%4==0 || answer%2!=0)
            {
                count+=1;
            }
            j++;
            if(j==n)
            {
                r++;i+=1;
                j=i;answer=1;
                if(r==n)
                {
                    break;
                }
            }
        }
        printf("%lld\n",count);
    }
    return 0;
}
```

File2:

```
#include<stdio.h>
int main()
{
    int i,j,t,n;
    long long int ans;
    long long int a[100000];
    long long int counter;
    scanf("%d",&t);
    while(t--)
    {
        counter=0;
        int ans=1,r=0;
        scanf("%lld",&n);
        for(i=0;i<n;i++)
        {
            scanf("%lld",&a[i]);
        }
        i=0;j=0;
        while(i<=j && j<n)
        {
            ans*=a[j];
            if(ans%4==0 || ans%2!=0)
            {
                counter+=1;
            }
            j++;
            if(j==n)
            {
                r++;i+=1;
                j=i;ans=1;
                if(r==n)
                {
                    break;
                }
            }
        }
        printf("%lld\n",counter);
    }
    return 0;
}
```

The two files look similar in terms of

1. Number of lines
2. Number of print statements, scan statements, integer variables etc
3. Also, number of loops, if statements, break statements also match.

- Considering all such aspects, the application collects data of both the files, and finds similarities among them.
- Finally, Cumulative matching percentage is calculated that represents the overall similarity of two files.
- Then, depending on the specified threshold, the Plagiarism is reported.

This way of detecting Plagiarism is so Efficient and Accurate.

Conclusion

Our project, titled 'Plagiarism Detector', aims to detect potential plagiarism in the given files.

Our algorithm is implemented on the given files, and the preliminary results of the comparison are presented to the user.

We intend to provide users a hassle-free way to detect plagiarism among the given files.

We hope our project contributes to the future anti-plagiarism engines and helps in combating plagiarism.

In future work, we may want to extend our comparison to a larger and more varied set of real-life data.

References

Internet Documents

Professional Internet Site

[1]. GeeksforGeeks | A computer science portal for geeks.

Available: https://www.geeksforgeeks.org/

Professional Internet Site

[2]. Stack Overflow – Where Developers Learn, Share, & Build Careers

Available: https://stackoverflow.com/questions/

Professional Internet Site

[3]. IBM Knowledge Center – Home of IBM product documentation

Available: https://www.ibm.com/support/knowledgecenter/

General Discussion forum

[4]. Tutorialspoint

Available: https://www.tutorialspoint.com/

Electronic Documents

Official C99 standard documentation

[5]. Available: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf

Official VS Code Documentation for configuring GCC on Linux

[6]. Available: https://code.visualstudio.com/docs/cpp/config-linux