IT PROJECT – GROUP 62

PLAGIARISM CHECKER

GROUP MEMBERS

ADITYA DILIP KAKAD (202052302)

AMAL SHAKYA (202052303)

AMAN VARSHNEY (202052304)

AMARTYA (202052305)

ANUBHAV KUSHWAHA (202052306)

ARCHIT AGRAWAL (202052307)

BRIEF EXPLANATION OF PROGRAM

- The Plagiarism Checker takes a number of files (< 100) and tells the percentage of data copied between every possible pair of those files.
- It takes a number of files, converts each file to a string, remove the usual characters from the string, makes a wordlist for each of the string.
- This wordlist contains the words in the sequential order present in the file.
- Now, it removes words (if opted) such as helping verbs, modal verbs etc.
- Now it compares the every possible pair of wordlists and tells the percentage of Plagiarism between the files.

ITS LOOK AT FRONT-END

- At first, a Home Screen appears, which provides four options –

  1. Check two files for Plagiarism 2. Check Multiple Files for Plagiarism

  3. Settings 4. Enter 0 to quit

- Choosing option 1 or 2 takes the user to a screen where it asks for file names. After giving the file names, it shows the Percentage plagiarism between the files.
- Choosing option 3 takes the user to a Settings Screen. On this screen, the user will see a list of category of words, each with an asterisk following it. The asterisk denotes that these words will be removed from the file during Plagiarism Check. The user can choose to keep these words during Plagiarism Check by inputting the index of the category of words. On doing so, the asterisk will be removed. User can return to Home Screen by entering 0.

CONSTRAINTS

- The Plagiarism Checker works only on text (.txt) files.
- The file must contain lesser than or equal to 7000 characters.
- The path of the files is fixed. A folder named “documents” should be present at the location where the code is being executed. The text files should be present in this “documents” folder.

For example, if the code is executed at “C:\Users\XYZ\Desktop\IT_Project.exe”, there should be a folder “documents” on the desktop which contains the text files.

CODE

```c
/* IT161 Project -: Group 62 -: Plagiarism Checker
Group Members are -
Aditya Dilip Kakad (202052302)
Amal Shakya (202052303)
Aman Varshney (202052304)
Amartya (202052305)
Anubhav Kushwaha (202052306)
Archit Agrawal (202052307)
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

const int MAX_CHARS = 7000;
const int MAX_WORDS = 2000;
const int MAX_WORD_LENGTH = 40;
const int MAX_FILE_NAME = 20;

const char *END_STR = "--WORDS--END--";
const char usual_chars[] = "?/.';[{]}=+-_`~!@#$%^&*()";
const char helping_verbs[][10] = {"is", "am", "are", "was", "were",
                                  "do", "does", "did", "has", "have", "had", "
be", "being", "been"};
const char modal_verbs[][10] = {"will", "shall", "would", "should", "can", "co
uld", "may", "might", "must"};
const char conjunctions[][10] = {"for", "or", "nor", "yet", "so", "and", "but"
, "if"};
const char preposition[][10] = {"as", "on", "of", "to", "by", "at", "up", "for
", "with", "down", "within", "without"};
const char adverb[][10] = {"now", "very", "never", "soon", "then", "much", "we
ll"};
const char article[][10] = {"a", "an", "the"};

int is_one_word_no_or_char = 1;
int is_helping_verbs = 1;
int is_modal_verbs = 1;
int is_conjunctions = 1;
int is_preposition = 1;
int is_adverb = 1;
int is_article = 1;

int contains_till_index(char word[], int lastIndex, char wordlist[][MAX_WORD_L
ENGTH]);

void readFile(FILE *file, char str[]);

void remove_usual_chars_and_lowercase(char usual_char[], char str[]);
```

```
void remove_usual_words(char wordlist[][MAX_WORD_LENGTH]);

void string_to_words(char str[], char wordlist[][MAX_WORD_LENGTH]);

double similarity(char wordlist1[][MAX_WORD_LENGTH], char wordlist2[][MAX_WORD
_LENGTH]);

void popout(int index, char wordlist[][MAX_WORD_LENGTH]);

int contains(char word[], char wordlist[][10]);

void similarity_meter(char file1name[], char file2name[], double value);

void clear_screen();

int homeScreen();

int getUserOptionHomeScreen(int start);

void settingsScreen();

int getUserOptionSettingsScreen(int start);

void twoFileScreen();

void multipleFileScreen();

void plagiarismCheck(char file1_name[], char file2_name[], double MinPercentag
e);

int main() {

    int option = homeScreen(); //function call, homeScreen() returns an intege
r
    switch (option) {
        case 1: {
            clear_screen();
            twoFileScreen(); // function call, takes to twoFileScreen
            break;
        }
        case 2:
            clear_screen();
            multipleFileScreen(); //function call, takes to multipleFileScreen
            break;
        case 3:
            clear_screen();
            settingsScreen(); //function call, takes to settingsScreen
```

```
            break;
        default:
            exit(0);
    }
    main();
    return 0;
}

void multipleFileScreen() {
    char files[100][MAX_FILE_NAME];

    printf("We Only Support 100 files at a time as of now\n");
    printf("** Enter 0 when you done entering all file Names **\n\n");

    int noOfFiles = 0;

    for (noOfFiles = 0; noOfFiles < 100;) {
        char tempFileName[20];
        printf("File %d : ", noOfFiles + 1);
        gets(tempFileName);   // for taking file name
        if (strcmp(tempFileName, "0") == 0) { // to stop taking more file name
s
            break;
        }

        if (strstr(tempFileName, ".txt") == NULL) { // to check if file contai
ns .txt
            strcat(tempFileName, ".txt");
        }

        FILE *tempFile;
        char filePath[100] = "documents\\";
        strcat(filePath, tempFileName);

        tempFile = fopen(filePath, "r");  // to open the file for reading
        if (tempFile == NULL) {            // If the file is not found
            printf("^^ This file can't be Found\n");
            continue;
        }
        fclose(tempFile); // close file
        strcpy(files[noOfFiles], tempFileName);
        noOfFiles++;
    }
    fflush(stdin); // to remove any buffer
    clear_screen();
    double minPercentage;
    printf("Enter Minimum Percentage of similarity in results\nex: 32 or 54.32
\n");
```

```
    scanf("%lf", &minPercentage);         // take minimum percentage of Plagia
rism from the user
    for (int i = 0; i < noOfFiles - 1; ++i) {
        for (int j = i + 1; j < noOfFiles; ++j) {
            plagiarismCheck(files[i], files[j], minPercentage); // to check Pl
agiarism for every pair of files.
        }
    }
    fflush(stdin); // to remove any buffer
    printf("Press Enter for Home Screen\n");
    getchar();
}

void twoFileScreen() {
    char file1_name[MAX_FILE_NAME];
    char file2_name[MAX_FILE_NAME];
    printf("Enter First File Name: ");
    gets(file1_name);                 // to get file 1 name
    printf("Enter Second File Name: ");
    gets(file2_name);                 // to get file 2 name
    plagiarismCheck(file1_name, file2_name, 0.0); // check Plagiarism function
    printf("\nPress Enter to Exit\n");
    fflush(stdin);    // to remove any buffer
    getchar();
    clear_screen();
}

void plagiarismCheck(char file1_name[], char file2_name[], double MinPercentag
e) {
    fflush(stdin);
    FILE *file1;
    FILE *file2;

    if (strstr(file1_name, ".txt") == NULL) { //checking if file name has .txt
 in it
        strcat(file1_name, ".txt");           // if file name does not contain
 .txt, it concatenates .txt to it
    }
    if (strstr(file2_name, ".txt") == NULL) {
        strcat(file2_name, ".txt");
    }

    char file1_path[100] = "documents\\"; //file 1 path
    char file2_path[100] = "documents\\"; //file 2 path

    strcat(file1_path, file1_name);  // to concatenate the file path and file
name
    strcat(file2_path, file2_name);
```

```
    file1 = fopen(file1_path, "r"); // opening file and using it as reading fi
le
    file2 = fopen(file2_path, "r");

    if (file1 == NULL || file2 == NULL) {
        printf("First File: %s\nSecond File: %s\n", file1_name, file2_name);
        printf("One or more files are not found.");
        return;
    }

    char file1str[MAX_CHARS]; // the text in the file will be contained in thi
s string
    char file2str[MAX_CHARS];

    readFile(file1, file1str); // function call, to convert the text in file i
nto string
    readFile(file2, file2str);

    fclose(file1); // closing the files
    fclose(file2);

    remove_usual_chars_and_lowercase(usual_chars, file1str); // function call,
 to remove usual characters as mentioned in Global Scope and to make every cha
racter lower case.
    remove_usual_chars_and_lowercase(usual_chars, file2str);

    char wordlist1[MAX_WORDS][MAX_WORD_LENGTH]; // this will contain a string
of words of the file1str
    char wordlist2[MAX_WORDS][MAX_WORD_LENGTH];

    string_to_words(file1str, wordlist1); //function call, to convert the stri
ng to string of words
    string_to_words(file2str, wordlist2);

    remove_usual_words(wordlist1); //function call, to remove usual words from
 the wordlist
    remove_usual_words(wordlist2);

    double a = similarity(wordlist1, wordlist2); //function call, to check the
 similarity between the files
    double value = acos(a);
    if (MinPercentage <= 100 - (value * 100.0) / (M_PI / 2)) {
        similarity_meter(file1_name, file2_name, value);
    }
    fflush(stdin);
}
```

```c
void settingsScreen() {
    printf(" Starred options will not be considered while finding similarity b
etween documents.\n\n");
    printf("  Toggle options by entering their index\n");
    printf("    1. [%c] Helping Verbs\n", (is_helping_verbs) ? '*' : ' ');
    printf("    2. [%c] Modal Verbs\n", (is_modal_verbs) ? '*' : ' ');
    printf("    3. [%c] Conjunctions\n", (is_conjunctions) ? '*' : ' ');
    printf("    4. [%c] Prepositions\n", (is_preposition) ? '*' : ' ');
    printf("    5. [%c] Adverbs\n", (is_adverb) ? '*' : ' ');
    printf("    6. [%c] Articles\n", (is_article) ? '*' : ' ');
    printf("    7. [%c] One letter word and no's\n", (is_one_word_no_or_char)
? '*' : ' ');
    printf("\nPress 0 for home screen\n");
    fflush(stdin);
    int option = getUserOptionSettingsScreen(1);
    if (option == 0) {
        return; // for returning to home screen
    }
    settingsScreen();
}

int homeScreen() {
    clear_screen();// function call, to print '- ' in one line and then leavin
g two blank lines
    printf("Welcome to Plagiarism Checker\n\n");
    printf("Enter 1, 2 or 3 to choose option\n");
    printf(" 1. Check Two Files for Plagiarism\n");
    printf(" 2. Check Multiple Files for Plagiarism\n");
    printf(" 3. Settings\n");
    printf("\nEnter 0 to quit\n");
    return getUserOptionHomeScreen(1); //function call, for the user to input
his/her choice.
}

int getUserOptionHomeScreen(int start) {
    char option;
    if (!start) {       // just in case if the user enters an invalid input
        printf("Sorry, please choose a valid option\n");
    }
    scanf("%c", &option);
    getchar(); // for caught '\n' when user enter a char
    fflush(stdin); // if user enter a string it will take first char and flush
 all input
    switch (option) {
        case '0':
            return 0;
        case '1':
            return 1;
```

```
        case '2':
            return 2;
        case '3':
            return 3;
        default:
            return getUserOptionHomeScreen(0);
    }
}

int getUserOptionSettingsScreen(int start) {
    char option;
    if (!start) { // just in case if the user enters an invalid input
        printf("Sorry, please choose a valid option\n");
    }
    scanf("%c", &option);  // option chosen by user.
    getchar();
    fflush(stdin); // if user enter a string it will take first char and flush
 all input
    // Each category of words are not to be considered in Plagiarism Check by
default
    // To toggle between the categories of words, user should input the index
of that category of words.
    switch (option) {

        case '0':
            return 0;
        case '1':
            if (is_helping_verbs) {
                is_helping_verbs = 0;
            } else {
                is_helping_verbs = 1;
            }
            return 1;
        case '2':
            if (is_modal_verbs) {
                is_modal_verbs = 0;
            } else {
                is_modal_verbs = 1;
            }
            return 2;
        case '3':
            if (is_conjunctions) {
                is_conjunctions = 0;
            } else {
                is_conjunctions = 1;
            }
            return 3;
        case '4':
```

```
            if (is_preposition) {
                is_preposition = 0;
            } else {
                is_preposition = 1;
            }
            return 4;
        case '5':
            if (is_adverb) {
                is_adverb = 0;
            } else {
                is_adverb = 1;
            }
            return 5;
        case '6':
            if (is_article) {
                is_article = 0;
            } else {
                is_article = 1;
            }
            return 6;
        case '7':
            if (is_one_word_no_or_char) {
                is_one_word_no_or_char = 0;
            } else {
                is_one_word_no_or_char = 1;
            }
            return 7;
        default:
            return getUserOptionSettingsScreen(0);
    }
}

void similarity_meter(char file1name[], char file2name[], double value) {
    int blockLength = 27;
    printf("File: %s and File %s\n", file1name, file2name);
    double percentageMatched = 100 - (value * 100.0) / (M_PI / 2);
    printf("Percentage Matched : %.2lf%%\n", percentageMatched);
    for (int i = 0; i < blockLength; ++i) {
        if (i < percentageMatched * (blockLength * 1.0 / 100)) {
            printf("|");
        } else {
            printf(":");
        }
    }
    printf("\n\n");
}

void readFile(FILE *filepo, char str[]) { // to convert the file into a string
```

```
    char c;
    c = fgetc(filepo);
    int i = 0;
    while (c != EOF && i < MAX_CHARS - 2) {
        str[i] = c;
        i += 1;
        c = fgetc(filepo);   // fgetc takes the next character of the file eve
ry time it is called.
    }
    str[i] = c;
    str[i += 1] = '\0'; // to end the string
}


void remove_usual_chars_and_lowercase(char usual_char[], char str[]) {
    char response[MAX_CHARS];
    int i = 0;
    int r = 0;
    while (str[i] != EOF) {
        char c = str[i];
        if (strchr(usual_char, c) == NULL) {
            if (c >= 65 && c <= 90) {
                c += 32; // make uppercase to lowercase
            }
            response[r] = c;
            r += 1;
        }
        i += 1;
    }
    response[r] = '\0'; // end the string
    strcpy(str, response); // copy response to str
}


double similarity(char wordlist1[][MAX_WORD_LENGTH], char wordlist2[][MAX_WORD
_LENGTH]) {
    int matches = 0;
    int wordlist1len = 0;
    int wordlist2len = 0;
    int count = 1;
    for (int i =0; i < MAX_WORDS && strcmp(wordlist1[i], END_STR) != 0; i++){
        if (contains_till_index(wordlist1[i], i, wordlist1)){
            popout(i, wordlist1);
        }
    }
    for (int i =0; i < MAX_WORDS && strcmp(wordlist2[i], END_STR) != 0; i++){
        if (contains_till_index(wordlist2[i], i, wordlist2)){
            popout(i, wordlist2);
        }
```

```
    }
    for (int i = 0; i < MAX_WORDS && strcmp(wordlist1[i], END_STR) != 0; ++i)
{
        wordlist1len++;
        for (int j = 0; j < MAX_WORDS && strcmp(wordlist2[j], END_STR) != 0; +
+j) {
            if (count == 1) {
                wordlist2len++;
            }
            if (!strcmp(wordlist1[i], wordlist2[j])) {
                matches++;       // to count number of words similar (in order
) in both the wordlists
            }
        }
        count = 0;
    }
    if (!matches) {
        return 0.0;
    }

    return (matches * 1.0) / (sqrt(wordlist2len * wordlist1len * 1.0)); // cal
culate cos(theta)
}


void string_to_words(char str[], char wordlist[][MAX_WORD_LENGTH]) {
    int i, j, cnt;
    j = 0;
    cnt = 0;
    int firstChar = 1;
    for (i = 0; i <= (strlen(str)); i++) {
        if (str[i] == ' ' || str[i] == ',' || str[i] == '\0') {
            wordlist[cnt][j] = '\0';
            cnt++;
            firstChar = 1;
            j = 0;
        } else {
            wordlist[cnt][j] = str[i];
            firstChar = 0;
            j++;
        }
    }
    strcpy(wordlist[cnt], END_STR);
}

void clear_screen() {
//    #ifdef WIN32
//        system("cls");
//    #else
```

```
//        system("clear");
//    #endif
    for (int i = 0; i < 30; ++i) {
        printf("- ");
    }
    printf("\n\n");
}

int contains_till_index(char word[], int lastIndex, char wordlist[][MAX_WORD_L
ENGTH]) {
    for (int i = 0; i < lastIndex; ++i) {
        if (!strcmp(word, wordlist[i])) {
            return 1;
        }
    }
    return 0;
}

void remove_usual_words(char wordlist[][MAX_WORD_LENGTH]) {

    for (int i = 0; i < MAX_WORDS && strcmp(wordlist[i], END_STR) != 0; ++i) {
        if (strlen(wordlist[i]) == 0) {
            popout(i, wordlist);
            i--;
            continue;
        }
        if (is_one_word_no_or_char) {
            if (strlen(wordlist[i]) == 1) {
                popout(i, wordlist);
                i--;
                continue;
            }
        }
        if (is_helping_verbs) {
            if (contains(wordlist[i], helping_verbs)) {
                popout(i, wordlist);
                i--;
                continue;
            }
        }
        if (is_modal_verbs) {
            if (contains(wordlist[i], modal_verbs)) {
                popout(i, wordlist);
                i--;
                continue;
            }
        }
        if (is_conjunctions) {
```

```
            if (contains(wordlist[i], conjunctions)) {
                popout(i, wordlist);
                i--;
                continue;
            }
        }
        if (is_preposition) {
            if (contains(wordlist[i], preposition)) {
                popout(i, wordlist);
                i--;
                continue;
            }
        }
        if (is_adverb) {
            if (contains(wordlist[i], adverb)) {
                popout(i, wordlist);
                i--;
                continue;
            }
        }
        if (is_article) {
            if (contains(wordlist[i], article)) {
                popout(i, wordlist);
                i--;
                continue;
            }
        }
    }

}

int contains(char word[], char wordlist[][10]) {
    int len = strlen(wordlist);
    for (int i = 0; i < len; ++i) {
        if (!strcmp(word, wordlist[i])) {
            return 1;
        }
    }
    return 0;
}



void popout(int index, char wordlist[][MAX_WORD_LENGTH]) {
    for (int i = index; strcmp(wordlist[i], END_STR) != 0; i++) {
        strcpy(wordlist[i], wordlist[i + 1]);
    }
}
```

EXPLANATION AND ALGORITHMS FOR THE CODE

The four pre-processor files that are included in the code are given below -

1. *#include <stdio.h>*
2. *#include <stdlib.h>*
3. *#include <string.h>*
4. *#include <math.h>*

- The functions printf(), scanf(), fopen(), fclose(), getchar(), gets(), fflush() belong to #include <stdio.h> pre-processor.
- The function exit() belongs to *#include <stdlib.h> pre-processor.*
- The functions strstr(), strcpy(), strcmp(), strlen(), strchr() and strcat() belong to #include<string.h> pre-processor.
- The function sqrt() and acos() belongs to #include<math.h> pre-processor.

The variables (given below) are declared as const int and hence their initialized values do not change throughout the program.

- MAX_CHARS = 7000 (this indicates that the file should have a maximum of 5000 characters)
- MAX_WORDS = 2000 (this indicates that the file should have a maximum of 2000 words)
- MAX_WORD_LENGTH = 40 (this indicates that each word in the file should be less than or equal to 40 characters)
- MAX_FILE_NAME = 20 (this indicates that the name of file should be of maximum 20 characters).

The const char *END_STR = "--WORDS--END--" is declared and its value doesn't change throughout the program. It is used to make a confirmation that the string has ended. That is, if the end of the string contains "--WORDS--END--", the string is ended.

- The string **const char usual_chars[] = "?/.';[{]}=+-_`~!@#$%^&*()"** contains of usual characters. These characters will be removed from the file before Plagiarism Check. As it can be seen, the string does not contain ' '(space) , ','(comma) because these characters will be used to separate words from the file string.

- The 2-D character array **const char helping_verbs[][10] = {"is", "am", "are", "was", "were" ,"do", "does", "did", "has", "have", "had", "be", "being", "been"}** contains helping verbs.
- The 2-D character array **const char modal_verbs[][10] = {"will", "shall", "would", "should", "can", "could", "may", "might", "must"}** contains modal verbs.
- The 2-D character array **const char conjunctions[][10] = {"for", "or", "nor", "yet", "so", "and", "but", "if"}** contains conjunctions.
- The 2-D character array **const char preposition[][10] = {"as", "on", "of", "to", "by", "at", "up", "for", "with", "down", "within", "without"}** contains prepositions.
- The 2-D character array **const char adverb[][10] = {"now", "very", "never", "soon", "then", "much", "well"}** contains adverbs.
- The 2-D character array **const char article[][10] = {"a", "an", "the"}** contains characters.

The usual characters will always be removed from the file before Plagiarism Check.

All the other categories of words mentioned above are removed by default during Plagiarism Check. However, an option is provided to retain User's Choice of Word Category to be kept in the file during Plagiarism Check.

The variables (given below) are all initialized to 1. This value can be changed later in the program.

- int is_one_word_no_or_char = 1;
- int is_helping_verbs = 1
- int is_modal_verbs = 1
- int is_conjunctions = 1
- int is_preposition = 1
- int is_adverb = 1
- int is_article = 1

If the value of these variables is 1, the corresponding category of words (as indicated by variable names) will be removed from the file and will not be considered in Plagiarism Check.

If the value of these variables is 0, the corresponding category of words (as indicated by variable names) will not be removed from the file and will be considered in Plagiarism Check.

The value of these variable can be changed on the Settings Screen during the execution of program by the User.

Algorithm for the function int main() -:

1. Declare an int variable option and initialize it with a function call homeScreen().
2. The integer returned by homeScreen() is stored in option.
3. If option is 1
    - Call the function clear_screen()
    - Call the function twoFilesScreen()
    - Go to step 7 using break statement
4. If option is 2
    - Call the function clear_screen()
    - Call the function multipleFilesScreen()
    - Go to step 7 using break statement
5. If option is 3
    - Call the function clear_screen()
    - Call the function settingsScreen()
    - Go to step 7 using break statement
6. If option is anything other than 1, 2 and 3 then exit(0).
7. Call main()
8. Return 0

Algorithm for the function void Read File(FILE *filepo, char str[]) -:

1. Initialize a character c as fgetc(filepo)
2. Initialize an integer i as 0.
3. While c is not equal to EOF and i < Max_CHARS – 2, do the following
    - Put i^{th} index of str as c.
    - Increase i by 1.
    - Put c = fgetc(filepo)
4. Put c in i^{th} index of str.
5. Put '\0' in (i + 1)th index of string str to end the string.

Algorithm for the function void remove_usual_chars_and_lowercase(char usual_char[], char str[])-:

1. Declare I = 0, r = 0 and char response*[MAX_CHARS]*
2. While the str[i] = !EOF, do the following steps
    - Declare a character "c" with value equal to str[i].
    - If string character is not an usual character, then do the following
        (i) If 65 <= c <= 90, then increase the value of c by 32 (to convert to uppercase from lowercase).
        (ii) Then equate, response[r] = c, then increase the value of r by 1. (r += 1).
    - Then increase the value of i by 1. *(i++)*
3. Then, equate response[r] = '\0' to end the string.
4. Copy the string response to string str.

Algorithm for the function void remove_usual_chars (char wordlist [] [MAX_WORD_LENGTH]) -:

1. Use a 'for' loop with i as counter variable. Initialize i = 0 and run the loop until either i < MAX_WORDS or wordlist[i] = END_STR, do the following

    - If string length of wordlist[i] is 0, do the following,

        (i) Call the function popout(i, wordlist)

        (ii) Decrease i by 1.

        (iii)Continue to next iteration of for loop using continue statement

    - If (one_word_no_or_char) i.e. if one words, numbers or characters are not to be considered in Plagiarism check.

        (i) If string length of wordlist[i] is 1, do the following

            a. Call the function popout(i, wordlist)

            b. Decrease i by 1.

            c. Continue to next iteration of for loop using statement.

- If (is_helping_verbs) i.e. if helping verbs are not to be considered in Plagiarism check.
    (i) If wordlist[i] contains helping_verbs
        a. Call the function popout (i, wordlist)
        b. Decrease i by 1.
        c. Continue to next iteration of for loop using continue statement.
- If (is_modal_verbs) i.e if modal verbs are not to be considered in Plagiarism Check
    (i) If wordlist[i] contains modal_verbs
        a. Call the function popout (i, wordlist)
        b. Decrease i by 1.
        c. Continue to next iteration of for loop using continue statement
- If (is_conjuctions) i.e. if conjuctions are not to be considered in Plagiarism Check.
    (i) If wordlist[i] contains conjuctions
        a. Call the function popout(i, wordlist)
        b. Decrease i by 1.
        c. Continue to next iteration of for loop using continue statement.
- If (is_preposition) i.e. if prepositions are not to be considered in Plagiarism Check.
    (i) If wordlist[i] contains preposition
        a. Call the function popout(i, wordlist)
        b. Decrease i by 1.

c. Continue to next iteration of for loop using continue statement.

- If (is_adverb) i.e. if adverbs are not to be considered in Plagiarism Check.

  (i) If wordlist[i] contains adverb

  a. Call the function popout(i, wordlist)

  b. Decrease i by 1.

  c. Continue to next iteration of for loop using continue statement.

- If (is_article) i.e. if articles are not to be considered in Plagiarism Check.

  (i) If wordlist[i] contains articles

  a. Call the function popout(i, wordlist)

  b. Decrease i by 1.

  c. Continue to next iteration of for loop using continue statement.

Algorithm for the function void string_to_words (char str[], char wordlist[][MAX_WORD_LENGTH] -:

1. Declare integer variables i, j, cnt and firstChar.
2. Initialize j = 0, cnt = 0, firstchar = 1.
3. Use a 'for' loop with i = 0 as initialization condition and until i < strlen(str) do the following
   - If there is a ' ' (space) or ','(comma) or '\0' at the i^{th} index of string, do the following
     (i) Put an end to the word using wordlist[cnt][j] = '\0'.
     (ii) cnt++

     (iii)Update firstChar = 1.

     (iv)Update j = 0.

- Else
  (i) Take the string str characters in wordlist[cnt][j]
  (ii) Update firstChar = 0
  (iii) j++

4. Copy the string END_STR to wordlist[cnt] so that last words in the wordlist recognise that the wordlist has ended.

Algorithm for the function double similarity(char wordlist1[][MAX_WORD_LENGTH], char wordlist2[][MAX_WORD_LENGTH]) **-:**

1. Declare three int variable matches, wordlist1len, wordlist2len, and initialize them to zero.
2. Declare another int variable count and initialize it to 1.
3. Make a for loop, initialized as i =0, until I < MAX_WORDS and strcmp(wordlist1[i], END_STR) is not equal to 0
   1. If contains_till_index(wordlist1[i], i, wordlist1) is non-zero, then call function popout(i, wordlist1)
4. Make a for loop, initialized as i =0, until I < MAX_WORDS and strcmp(wordlist2[i], END_STR) is not equal to 0
   1. If contains_till_index(wordlist2[i], i, wordlist2) is non-zero, then call function popout(i, wordlist2)
5. Use a for loop which is initialized as int i = 0, until If i < MAX_WORDS and strcmp(wordlist1[i], END_STR) is not equal to 0.
   - Increase wordlist1len by 1.
   - Use another for loop in the body of above for loop which is initialized as j = 0, run until I > Max_WORDS and strcmp(wordlist2[j], END_STR) is not equal to 0.
     (i) If (count ==1) is true, increase wordlist2len by 1.
     (ii) If string at ith index of wordlist1 and jth index of wordlist 2 are same then increase matches by 1.
   - Update count to 0.
6. If (!matches) is true then return 0.0;
7. Return (matches*1)/(sqrt(wordlist2len*wordlist1len*1.0)); as value of cos theta.

Algorithm for the function void popout(int index, char wordlist[][MAX_WORD_LENGTH]) *-:*

1. Use a for loop with initialization int i = index, until the i^{th} index of wordlist (i.e wordlist[i]) is not same as END_STR
    - Overwrite the i^{th} index element by copying the string wordlist[i + 1] to wordlist[i].

Algorithm for the function int contains(char word[], char wordlist[][10]) *-:*

1. Declare len as string length of wordlist [strlen(wordlist)]
2. For i =0, until (i < len) holds true, do the following
    - If the word and ith index of wordlist are equal, it returns 1
    - Update i by i + 1
3. Return zero.

Algorithm for the function void similarity_meter (char file1name[], char file2name[], double value) -:

1. Declare an int variable blockLength and initialize it 27.
2. Print the name of the files.
3. Declare a double variable percentageMatched
4. percentageMatched = 100 – (value * 100)/ (M_PI/2)
5. Print percentageMatched
6. For integer i = 0 to i less blockLength, perform following steps ***(this is just for presentation)***
    - If i is less than (percentageMatched * (blockLength * 1.0/ 100), print '|'.
    - Else print ':'
7. Leave two lines blank.

Algorithm for the function void clear_screen () -:

1. Print hyphen and space ("- ") 30 times in a line.
2. Leaves two lines blank.

Algorithm for the function int homeScreen() -:

1. Call the function clear_screen().
2. Print
   "Welcome to Plagiarism Checker.
   Enter 1, 2 or 3 to choose option

    1. Check Two Files for Plagiarism

2. Check Multiple Files for Plagiarism
3. Settings

Enter 0 to quit”

3. Call the function getUserOptionHomeScreen(1)
4. Return the value obtained from function call in Step 4.

Algorithm for the function int getUserOptionHomeScreen(int start) -:

1. If (!start) is true print “Sorry, please choose a valid option”
2. Take a char input from the user and store it in ‘option’ variable (only options 0, 1, 2, 3 are valid).
3. If input from user is 0, return 0.
4. If input from user is 1, return 1.
5. If input from user is 2, return 2.
6. If input from user is 3, return 3.
7. If user input anything other than the options 0, 1, 2, 3, it will return getUserOptionHomeScreen(0) which will basically execute the Step 1 of this function and next steps of this function and ask for a valid input.

Algorithm for the function void settingsScreen() -:

1. Print
   “ Starred options will not be considered while finding similarity between documents.
   Toggle options by entering their index”
2. It will check which category of words is to be considered and which is not to be and print ‘*’ or ‘ ‘ accordingly.
   For example,
   “[*] Helping Verb” will be printed if helping verbs are not to be considered in Plagiarism Check while it will print “[ ] Helping Verb” is helping verbs are to be considered in Plagiarism Check.
3. This will be checked and accordingly printed for every category of words.
4. Print “Press 0 for home screen”.
5. Declare an int variable option and initialize it with the function call getUserOptionSettingsScreen(1)
6. getUserOptionSettingsScreen(1) will return an integer to the variable option.

7. If option is 0, simply return from the function settingsScreen (to go to Home Screen), otherwise call settingsScreen() (to show the changes that user has made in Settings).

Algorithm for the function int getUserOptionSettingsScreen(int start) -:

Note - All the helping verbs, modal verbs, conjuctions, prepositions, adverbs and article and one letter words and no's will be removed from the file before Plagiarism Check by default. In the settings screen this is represented by a '*' every type of word. If the user does not want to remove any kind of words, he/she may choose using the toggle options provided in this function.

1. If (!start) is true print "Sorry, please choose a valid option".
2. Take a char input from user and store it in 'option' variable (only options 0, 1, 2, 3, 4, 5, 6, 7 are valid)
   Each Valid input denotes a category of words that are mentioned in the table below.

Option	Category of Words
0	No category of words, it is used to exit the settings screen
1	Helping Verbs
2	Modal Verbs
3	Conjuctions
4	Prepositions
5	Adverb
6	Article
7	1 word or number or single char

3. If input option is 0, the setting screen will exit to home screen.
4. Except for 0, if any other valid option is chosen, it will consider the corresponding type of words if they were not considered for Plagiarism Check before and vice-versa. Afterwards it will return integer corresponding to that type of words (so that change is visible to the user on the Settings Screen).
5. If an invalid option is chosen, it will return getUserOptionSettingsScreen(0) which will basically execute the Step 1 of this function and next steps of this function and ask for a valid input.

Algorithm for the function void twoFileScreen () -:

1. Declare two character arrays which holds the name of the files to be checked.
2. Call the function plagiarismCheck(file1_name, file2_name, 0.0)
3. After returning from the function plagiarismCheck, print “Press Enter to exit).
4. Call the function clear_screen ().

***Algorithm for the function void multipleFileScreen()*-:**

1. Initialize a 2-D char array to store names of the files.
2. Print “We Only Support 100 files at a time as of now”
   Print “** Enter 0 when you done entering all file Names **”
3. For noOfFiles = 0 until noOfFiles < 100:
    - Initialize a character array to store file name.
    - Take the File name by the user using gets function.
    - If file name equals to 0 then break the loop.
    - If file name does not contain “.txt” then concatenate .txt extension to the file name.
    - Initialize a char array filepath equals to “documents\\”
    - Concatenate file name to filepath
    - Open the file in read mode
    - If file don’t exist, print “^^ This file can't be Found” and continue.
    - Close the file
    - Add the file name file names at index noOfFiles
    - Increase noOfFiles by 1
4. Flush any previous buffer.
5. Call function clear_screen().
6. Declare minPercentage.
7. Print “Enter Minimum Percentage of similarity in results
   ex: 32 or 54.32”.
8. Take double input to minPercentage.
9. For integer i = 0, until i < noOfFiles – 1
    - For j = i + 1, until j < noOfFiles
        i. Call function plagiarismCheck(file1_name, file2_name, minPercentage)
        ii. Increase j by 1.

- Increase i by 1

10. Flush any previous buffer.
11. Printf “Press Enter for Home Screen”.
12. Call function getchar().

Algorithm for the function void plagiarismCheck (char file1_name[], char file2_name, double MinPercentage) -:

1. Clear buffer input, if any. The function fflush is called to remove any previous buffer.
2. Check if file names consist ‘.txt’. If not, the concatenate it in the filenames.
3. Declare char arrays file1_path and file2_path and enter the path of the file. (Documents is entered by default, its better if the user save the files in this path only).
4. Concatenate the file path and file name for both files.
5. Open the files using fopen and use it for reading only.
6. If file1 or file2 is Null, print “One or more files are not found.” and return.
7. Declare two character arrays file1str[MAX_CHARS] and file2str[MAX_CHARS].
8. Call the function void readFile(file1, file1str) for both the files one by one. The purpose for this function call is to convert the text file into a string.
9. Use library function fclose to close both the files one by one.
10. Call the function void remove_usual_chars_and_lowercase(usual_char, file1str) for both the strings one by one. The purpose of this function call is to remove the usual characters and to convert the whole string in lowercase letters.
11. Declare two 2-D character arrays wordlist1[MAX_WORDS][MAX_WORD_LENGTH] and wordlist2[MAX_WORDS][MAX_WORD_LENGTH]
12. Call the function void string_to_words(file1str, wordlist1) for both the strings one by one. The purpose of this function call is to convert the string into words.
13. Call the function void remove_usual_words (wordlist1) for both the wordlists one by one. The purpose of this function call is to remove the category of words that are not to be considered in Plagiarism Check.

14. Declare a double variable a and initialize it with the function call double similarity(wordlist1 , wordlist2). The purpose of this function call is to get the extent of similarity between the two wordlists in terms of cosine.
15. Declare a double variable value and

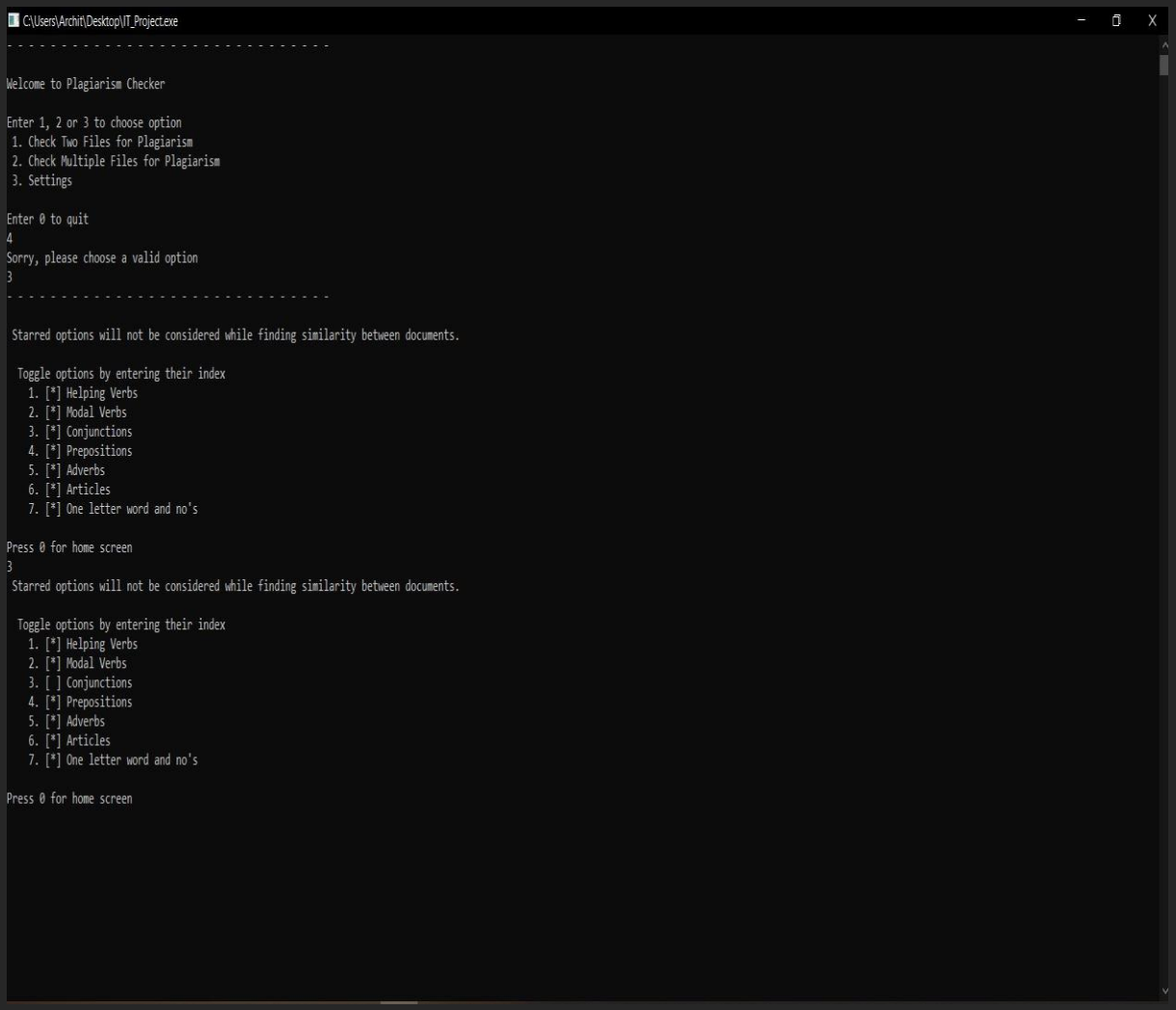
value = acos (a)

16. If percentage of plagiarism is greater than equal to the minimum value of plagiarism percentage entered by the user, call the function similarity_meter(file1_name, file2_name, value)
17. fflush to remove any buffer.

Algorithm for the function void contains_till_index (char word[], int lastIndex, char wordlist[][MAX_WORD_LENGTH]) -:

1. For i =0, until (i < lastIndex) holds true, do the following
    - If the word and ith index of wordlist are equal, it returns 1
    - Update i by i + 1
2. Return zero.

SAMPLE OUTPUT


C:\Users\Archit\Desktop\IT_Project.exe
- -
Welcome to Plagiarism Checker
Enter 1, 2 or 3 to choose option
1. Check Two Files for Plagiarism
2. Check Multiple Files for Plagiarism
3. Settings
Enter 0 to quit
4
Sorry, please choose a valid option
3
- -
Starred options will not be considered while finding similarity between documents.
Toggle options by entering their index
1. [*] Helping Verbs
2. [*] Modal Verbs
3. [*] Conjunctions
4. [*] Prepositions
5. [*] Adverbs
6. [*] Articles
7. [*] One letter word and no's
Press 0 for home screen
3
Starred options will not be considered while finding similarity between documents.
Toggle options by entering their index
1. [*] Helping Verbs
2. [*] Modal Verbs
3. [ ] Conjunctions
4. [*] Prepositions
5. [*] Adverbs
6. [*] Articles
7. [*] One letter word and no's
Press 0 for home screen

```
C:\Users\Archit\Desktop\IT_Project.exe

Press 0 for home screen
9
Sorry, please choose a valid option
6
 Starred options will not be considered while finding similarity between documents.

  Toggle options by entering their index
    1. [*] Helping Verbs
    2. [*] Modal Verbs
    3. [ ] Conjunctions
    4. [*] Prepositions
    5. [*] Adverbs
    6. [ ] Articles
    7. [*] One letter word and no's

Press 0 for home screen
0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Welcome to Plagiarism Checker

Enter 1, 2 or 3 to choose option
 1. Check Two Files for Plagiarism
 2. Check Multiple Files for Plagiarism
 3. Settings

Enter 0 to quit
1
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Enter First File Name: testFile1
Enter Second File Name: testFile2
File: testFile1.txt and File testFile2.txt
Percentage Matched : 44.98%
||||||||||||::::::::::::


Press Enter to Exit

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
C:\Users\Archit\Desktop\IT_Project.exe
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Welcome to Plagiarism Checker

Enter 1, 2 or 3 to choose option
 1. Check Two Files for Plagiarism
 2. Check Multiple Files for Plagiarism
 3. Settings

Enter 0 to quit
2
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

We Only Support 100 files at a time as of now
** Enter 0 when you done entering all file Names **

File 1 : testFile1
File 2 : testFile3
File 3 : testFile5
File 4 : 0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Enter Minimum Percentage of similarity in results
ex: 32 or 54.32
25
File: testFile1.txt and File testFile3.txt
Percentage Matched : 78.77%
||||||||||||||||||||:::::

File: testFile3.txt and File testFile5.txt
Percentage Matched : 69.46%
|||||||||||||||||:::::::

Press Enter for Home Screen

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Welcome to Plagiarism Checker

Enter 1, 2 or 3 to choose option
 1. Check Two Files for Plagiarism
 2. Check Multiple Files for Plagiarism
 3. Settings

Enter 0 to quit
```

REFERENCES

- https://www.geeksforgeeks.org/measuring-the-document-similarity-in-python/

  This gave us the idea of making a Plagiarism Checker.

- https://www.knowprogram.com/c-programming/2d-array-of-strings-in-c/

  Learnt about 2-D array of strings.

- https://fresh2refresh.com/c-programming/c-function/c-library-functions/

  Learnt about the library functions.

- Youtube Channel 'Code With Harry' and 'LetUsC' by Yashwant Kanetkar for studying the concept of strings.

SAMPLE TEXT FILES

Sample Text 1

C is a procedural programming language developed by Dennis Ritchie between 1969 and 1973. It was mainly developed as a system programming language to write operating systems.

The main features of C include low – level access to memory, simplest set of keywords, and clean style. These features make C language suitable for system programming

like operating systems or compiler development.

Despite the generality of high – level languages, C continues to empower the world. Some systems that are used by millions of people and are programmed in C are mentioned below -:

Microsoft Windows, Linux, OS X, iOS, Android and Windows Phone kernels are written mostly in C.

Databases including Oracle, MySQL and PostgreSQL are coded in C (and C++).

Learning C has a lot of benefits but the foremost thing is that it helps us to understand the underlying architecture of how things work.

Consider a situation in which a man has to buy a car. What he can do is he can buy a car with advanced technological features such as auto-driving mode

and auto gear changing. But if he buys a car which does not provide him these features, he will learn how these things actually works. For a person who is comfortable

with a manual car will easily adapt an automated car, but for a person who only knows how to drive an automated car will take some time to switch to a manual car.

Similarly, if a person learns C programming language first, it will help him to learn other high-level languages as it helps him/her to understand the underlying

architecture of operating systems.

Some important advantages of learning C programming language

C is a middle level language. The middle-level languages are somewhere between the Low-level machine understandable assembly languages and High-Level

user-friendly languages. Being a middle- level language, it is as close to the machine language as possible. It reduces the gap between low-level and high-level languages.

Helps us to understand the fundamentals of Computer Theories. Most of the theories related to computers like Computer Networks, Compiler Designing,

Computer Architecture and Operating systems are based on C programming language. In the modern high-level languages, the machine level details are hidden

from the user, so in order to work with CPU cache, memory, network adapters, learning C programming is a must.

Fewer Libraries – C has fewer libraries in comparison to modern high-level languages. Hence, learning C clears the concepts of programming to a great extent

as you have to write a lot of things from scratch rather than using a library function for it.

C is very fast in terms of execution time. Programs written and compiled in C executes much faster than compared to any other programming language. C programming

language is very fast in terms of execution as it does not have any additional processing overheads such as garbage collection or preventing memory leaks etc.

The programmer must take care of these things on his own.

Memory Manipulation – Arbitrary memory address access and pointer arithmetic is an important feature that makes C a perfect fit for system programming

(operating systems and compiler designing).

C versus other languages

C versus C++ -: C++ is an extension of C. While still being C-like in syntax and approach, C++ provides many genuinely useful features that aren’t

available natively in C; namespaces, templates, exceptions, automatic memory management and so on.

C versus Java -: Java syntax borrows a great deal from C and C++. Unlike C, though, Java doesn't by default compile to native code. The "write once, run anywhere"

philosophy behind Java also allows Java programs to run with relatively little tweaking for a target architecture. By contrast, although C has been ported to a

great many architectures, any given C program may still need customization to run properly on, say, Windows versus Linux.

C versus Python -: Python emphasizes on speed of development rather than speed of execution. A program that may take an hour to be assembled in C can be

assembled in minutes using Python while that program may take seconds to execute in C, but a minute to run in Python.

Another major difference in memory management. Python programs are fully memory managed by Python runtime, so developers don't have to worry about

allocating and freeing memory. On the other hand, writing C programs requires tremendous attention to memory management.

Sample Text 2

The C programming language came out of Bell Labs

in the early 1970s. The development of the C

language by Dennis Ritchie.

What is C language?

C is a general – purpose programming

language that is extremely popular, simple , and flexible to

use. It is structured programming language that is machine

independent and extensively used to write various

application , Operating System like windows , and many

other complex programs like Oracle database , Git , Python

interpreter , and more.

Its Importance :

There are plenty reasons to believe that C

programming has remain active for a long time and will also.

There are many programming language that allow

developers to be more productive that with c for different

kinds of projects. Here are some reasons that c is

unbeatable , and almost mandatory , for certain

applications.

C is a great language for expressing common ideas inprogramming in a way that most people are comfortable

with.

C is almost a portable assembly language. It is as close

to the machine as possible while it is almost universally

available for existing processor architecture.

A lot of the principle used in C for instance and for

command line parameters as well as loop constructs

and variable types.

C remains the most popular programming language

language for programming microcontrollers in

embedded systems.

C compilers can produce highly efficient code.

There are only 32 keywords; several standard function

are available which can be used for developing

programs

C is portable language; this means that c programs

written for one computer can be run on another system,

with little or no modification.

C language is small and relatively easy to learn. Its

mandatory to learn C for beginners.

C has been used to implement several major operating

systems and Kernels , including Unix , Linux , MacOS ,

and Windows.C language has its ability to extend itself.

Sample Text 3

Strength of c programming language over other languages and reason for its relevance.

good reasons for that. Many of the C projects that exist today were started decades ago.

The UNIX operating system's development started in 1969, and its code was rewritten in C in

1972. The C language was actually created to move the UNIX kernel code from assembly to a

higher level language, which would do the same tasks with fewer lines of code.

Oracle database development started in 1977, and its code was rewritten from assembly to C

in 1983. It became one of the most popular databases in the world.

In 1985 Windows 1.0 was released. Although Windows source code is not publicly available,

it's been stated that its kernel is mostly written in C, with some parts in assembly. Linux

kernel development started in 1991, and it is also written in C. The next year, it was released

under the GNU license and was used as part of the GNU Operating System. The GNU

operating system itself was started using C and Lisp programming languages, so many of its

components are written in C.

But C programming isn't limited to projects that started decades ago, when there weren't as

many programming languages as today. Many C projects are still started today; there are

some

How is the World Powered by C?

Despite the prevalence of higher-level languages, C continues to empower the world. The

following are some of the systems that are used by millions and are programmed in the C

language.Linux is also written mostly in C, with some parts in assembly. About 97 percent of

the world's 500 most powerful supercomputers run the Linux kernel. It is also used in many

personal computers.

Mac Mac computers are also powered by C, since the OS X kernel is written mostly in C. Every

program and driver in a Mac, as in Windows and Linux computers, is running on a C powered kernel.

Mobile

iOS, Android and Windows Phone kernels are also written in C. They are just mobile

adaptations of existing Mac OS, Linux and Windows kernels. So smartphones you

use every day are running on a C kernel.

Databases.

The world’s most popular databases, including Oracle Database, MySQL, MS SQL

Server, and PostgreSQL, are coded in C (the first three of them actually both in C and

C++).

Databases are used in all kind of systems: financial, government, media,

entertainment, telecommunications, health, education, retail, social networks, web,

and the like.

3D Movies

3D movies are created with applications that are generally written in C and C++.

Those applications need to be very efficient and fast, since they handle a huge

amount of data and do many calculations per second. The more efficient they are, the

less time it takes for the artists and animators to generate the movie shots, and the

more money the company saves.

Embedded Systems Imagine that you wake up one day and go shopping. The alarm clock that wakes you

up is likely programmed in C. Then you use your microwave or coffee maker to make

your breakfast. They are also embedded systems and therefore are probably

programmed in C. You turn on your TV or radio while you eat your breakfast. Those

are also embedded systems, powered by C. When you open your garage door with the

remote control you are also using an embedded system that is most likely

programmed in C.

Then you get into your car. If it has the following features, also programmed in C:

automatic transmission

tire pressure detection systems

sensors (oxygen, temperature, oil level, etc.)

memory for seats and mirror settings.

dashboard display

anti-lock brakes

automatic stability control

cruise control

climate control

Sample Text 4

Brief History and Introduction:

A great computer scientist named “Dennis Ritchie” created a programming language called 'C' at the Bell Laboratories in 1972. It was created from 'ALGOL', 'BCPL' and 'B'

programming languages which were created before ‘C’. 'C' programming language contains all the features of these languages and many more additional concepts and features

that make it unique from other languages.

'C' is a powerful programming language and it is strongly associated with the UNIX operating system and even most of the UNIX operating system is coded in 'C'. Initially

at starting 'C' programming was limited to the UNIX operating system, but as it started spreading around the world, it became commercial, and many compilers were released

for cross-platform systems.

Now a day’s 'C' runs under a variety of operating systems and hardware platforms. As it was started evolving, many different versions of the language were released.

In past, At times it became difficult for the developers to keep up with the latest version as the systems were running under the older versions. To assure that it will

remain standard, American National Standards Institute (ANSI) defined a commercial standard for 'C' language in 1989. Later, it was approved by the International

Standards Organization (ISO) in 1990. 'C' programming language is also called as 'ANSI C'.

Languages such as C++/Java were developed from 'C'. These languages are widely used in various technologies and applications. Thus, 'C' forms a base for many other

programming languages that are currently in use.

Strength of C programming language:

C is a powerful and more efficient language as it contains many data types and operators to perform all kind of operations. It is widely used in embedded systems also.

C is very flexible or we can say it is machine independent that helps you to run your code on any machine without making any change or just a few changes in your code.

A crucial ability of C is to extend itself. It has its own sets of built in functions in the C library, therefore it becomes easy to use these functions. We can also

add our own functions to the C standard library and makes code simpler.

C is structure based. It means that the issues or complex problems can be divided into smaller blocks or functions. This modular structure of ‘C’ helps in easier and

simpler testing and maintenance.

'C' contains 32 keywords, various data types and a set of powerful built-in functions that make programming very efficient for user.

C is a middle level programming language that means it supports high-level programming as well low level programming language. It supports the use of kernels and drivers

in low programming and also supports system software applications in the high level programming language.

The use of algorithms and data structures in C has made program computations very fast and smooth. Thus, the C language can be used in the complex calculations and

operations such as MATLAB.

C language is case-sensitive which means lowercase and uppercases letters are treated differently in C language.

C provides dynamic memory allocation that allows you to allocate memory at run time. For example, if you don’t know how much memory required by objects in your

program, you can still run a program in C and assign the memory at the same time.

C is very portable language. Different components of windows, UNIX, and Linux systems are written in C and many other complex programs like Oracle database, Git,

python interpreter and more also written in C.

C is a general purpose language, so we can use C for game design, graphics, enterprise applications etc.

C is a base for the computer programming. If you know 'C’ then it is easy for you to easily grasp the knowledge of the other programming languages that uses the concepts of 'C'.

Sample Text 5

Importance of C programming language over other programming languages:

C is a middle level programming language developed by Dennis Ritchie and Brian Kernighan in 1972 at the AT&T Bell Labs in the USA.

The objective of its development was in the context of the re-design of the UNIX operating system so that it can be used on multiple computers.

This language is now 49 years old but still its importance hasn't reduced over the course of time and it is recommended for all the new programmers to start with C.

As a middle-level language, C combines the features of both high-level and low-level languages.

It is robust language whose rich setup of built in functions and operator can be used to write any complex program.

It can be used for low-level programming, such as scripting for drivers and kernels and it also supports functions of high-level programming

languages, such as scripting for software applications etc.

C is a structured programming language which allows a complex program to be broken into simpler programs called functions.

It also allows free movement of data across these functions.Various features of C including direct access to machine level hardware APIs,

the presence of C compilers, deterministic resource use and dynamic memory allocation make C language an optimum choice for scripting applications and drivers of embedded systems.

C language is case-sensitive which means lowercase and uppercase letters are treated differently.

C is highly portable and is used for scripting system applications which form a major part of Windows, UNIX, and Linux operating system.

C is a general-purpose programming language and can efficiently work on enterprise applications, games, graphics, and applications requiring calculations, etc.

C language has a rich library which provides a number of built-in functions. It also offers dynamic memory allocation.

Program written in C are efficient due to several variety of data types and powerful operators.

C implements algorithms and data structures swiftly, facilitating faster computations in programs. This has enabled the use of C in applications

requiring higher degrees of calculations like MATLAB and Mathematica.

The C compiler combines the capabilities of an assembly language with the feature of high level language.

Therefore it is well suited for writing both system software and business package.

There are only 32 keywords; several standard functions are available which can be used for developing program.

C is portable language; this means that C programs written for one computer system can be run on another system, with little or no modification.

C language is well suited for structured programming, this requires user to think of a problems in terms of function or modules or block.

A collection of these modules make a program debugging and testing easier.