Assignment No. 03

Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define check(ch)((ch >= 65 && ch <= 91) || (ch >= 97 && ch <= 123))
typedef struct invert_table
{
	char vocab[30];
	char occur[30][10];
	int total;
}

table;
void display(table mat[30], int size)		//Display inverted index table
{
	int i, j, k;
	printf("\nVocabulary\t Occurance");
	for (i = 0; i < size; i++)
	{
		printf("\n%10s\t", mat[i].vocab);
		for (j = 0; j < mat[i].total; j++)
		{
			printf(" %4s ", mat[i].occur[j]);
		}
	}
}

char *my_itoa(int num, char *str)		//convert integer to string
{
	if (str == NULL)
		return NULL;
	sprintf(str, "%d", num);
	return str;
}

int insert_voc(table mat[30], int size, char str[20], int pos, int num)
{
	char string[20], s1[20];
	int i, m, n;
	for (m = 0; m < size; m++)
	{
		if (strcmp(mat[m].vocab, str) == 0)		//check if word is already present
		{
			my_itoa(num, s1);
			strcpy(string, "D");
			strcat(string, s1);		//D1
			strcat(string, ".");		//D1.
			my_itoa(pos, s1);		//pos
			strcat(string, s1);		//D1.pos
			n = mat[m].total;
			strcpy(mat[m].occur[n], string);
```

```
                        mat[m].total++;//increment count
                        break;
                }
        }

        if (m == size)
        {
                size++;
                strcpy(mat[m].vocab, str);
                mat[m].total = 0;
                my_itoa(num, s1);
                strcpy(string, "D");
                strcat(string, s1);
                strcat(string, ".");
                my_itoa(pos, s1);
                strcat(string, s1);
                n = mat[m].total;
                strcpy(mat[m].occur[n], string);
                mat[m].total = 1;
        }

        return size;
}

int inverted_file(table t[20], char str_file[20], int num, int size)
{
        FILE *fp = NULL;
        char ch, str[20];
        int i = 0, pos, cnt = 0;
        fp = fopen(str_file, "r");
        if (fp == NULL)
                printf("Error");
        else
        {
                do {    ch = getc(fp);
                        cnt++;  //increment position
                        if (ch != ' ')        //store word
                        {
                                if (i == 0)
                                        pos = cnt;
                                str[i++] = ch;
                        }
                        else
                        {
                                if (str[i - 1] == '.')
                                        i--;
                                str[i] = '\0';
                                size = insert_voc(t, size, str, pos, num);
                                i = 0;
                                pos = 0;
                        }
                } while (ch != EOF);
        }
```

```
	fclose(fp);
	return size;
}

void main()
{
	FILE *fp, *fp1;
	char fip[15], fop[15], c, word[20];
	int i = 0, j, k, size = 0, number;
	table t[20];
	printf("\nEnter the number of files ");
	scanf("%d", &number);

	for (k = 1; k <= number; k++)	//iterate for every file
	{
		printf("\nEnter name of input file : ");
		scanf("%s", fip);
		printf("\nEnter name of output file : ");
		scanf("%s", fop);
		fp = fopen(fip, "r");
		fp1 = fopen(fop, "w+");
		while ((c = getc(fp)) != EOF)	//write words in output file
		{
			if (c != ' ')
			{
				word[i] = c;
				i++;
			}
			else
			{
				if (word[i - 1] == '.')
					i--;
				word[i] = '\0';
				fprintf(fp1, "%s", word);
				fprintf(fp1, "\n");
				for (j = 0; j <= i; j++)
					word[j] = '\0';
				i = 0;
			}
		}

		size = inverted_file(t, fip, k, size);
	}	//for
	printf("\n----------The Inverted Index Table is----------\n");
	display(t, size);
}
```

Output:

Output1 –
ubuntu@ubuntu:~/Desktop$ gcc inverted.c
ubuntu@ubuntu:~/Desktop$ ./a.out

Enter the number of files 3
Enter name of input file : ip1
Enter name of output file : op11
Enter name of input file : ip2
Enter name of output file : op12
Enter name of input file : ip3
Enter name of output file : op13

----------The Inverted Index Table is----------
Vocabulary Occurance
Inverted D1.1 D2.29
file D1.10 D2.38
algorithm D1.15 D1.88 D1.143 D2.43 D3.1 D3.82
implemented D1.28 D3.14
project D1.48 D3.34
necessary D1.63
optimizing D1.77
code D1.98
faced D1.106 D3.45
problem D1.117 D3.56
implementing D1.130 D2.16 D3.69
enjoyed D2.3
Task D3.93
assigned D3.101
student D3.113
completion D3.125
manual D3.139

Input1 file –
Inverted file algorithm is implemented in this project. It is necessary for optimizing algorithm code. I faced some problem when implementing algorithm.

Input2 file –
I enjoyed when implementing Inverted file algorithm.

Input3 file –
algorithm is implemented in this project. I faced some problem when implementing algorithm. Task is assigned to student for completion of manual.

Output1 file –
Inverted
file
algorithm
implemented
project
necessary
optimizing

algorithm
code
faced
problem
implementing
algorithm

Output2 file –
enjoyed
implementing
Inverted
file
algorithm

Output3 file –
algorithm
implemented
project
faced
problem
implementing
algorithm
Task
assigned
student
completion
manual