command line arguments:

\*command line arguments are supplied in main

ex:main(int argc=the number of arguments,char \*\*argv[](or)\*argv[]=which are the arguments)

\*Arguments will be consider as strings

=>#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc,char \*argv[])

{

printf("\nNo of args=%d",argc);=>to print no of arguments passed

printf("\n\n");

return 0;

}

=>gcc -Wall -g filename -o app

=>./app arguments

=>#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc,char \*argv[])

{

int i;

printf("\nNo of args=%d",argc);

for(i=0;i<argc;i++);=>to get the arguments passed

{

printf("\nargv[%d]=%s",i,argv[i]);

}:

printf("\n\n");

return 0;

}

=>Write a program to read the contents of the file and store them in the employee

records respectively using command line arguments to supply the databse

// employee.h

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

#define BUFF 1024

// Structure to store employee data

typedef struct {

int id;

char Name[20];

char Gender;

int Phno;

int salary;

} emp;

// Function declarations

int toreadfromfile(char \*fileName, emp employees[]);

void display\_arguments(int argc, char \*argv[]);

void display\_employee\_data(emp employees[], int count);

#endif // EMPLOYEE\_H

// employee.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "employee.h"

// Function to read data from file and process it

int toreadfromfile(char \*fileName, emp employees[]) {

FILE \*fp;

int count = 0;

char buffer[BUFF];

fp = fopen(fileName, "r");

if (fp == NULL) {

perror("Error opening file");

exit(1);

}

printf("\nOpened the file successfully\n");

// Read the file line by line and tokenize it

while (fgets(buffer, sizeof(buffer), fp)) {

char \*token = strtok(buffer, ",");

if (token != NULL) {

employees[count].id = atoi(token); // Convert the ID to integer

}

token = strtok(NULL, ",");

if (token != NULL) {

strncpy(employees[count].Name, token, sizeof(employees[count].Name) - 1);

}

token = strtok(NULL, ",");

if (token != NULL) {

employees[count].Gender = token[0]; // Assume Gender is a single character

}

token = strtok(NULL, ",");

if (token != NULL) {

employees[count].Phno = atoi(token); // Convert phone number to integer

}

token = strtok(NULL, ",");

if (token != NULL) {

employees[count].salary = atoi(token); // Convert salary to integer

}

count++;

if (count >= BUFF) {

printf("Maximum number of employees reached.\n");

break;

}

}

fclose(fp);

return count; // Return the number of employees read

}

// Function to display arguments passed to the program

void display\_arguments(int argc, char \*argv[]) {

printf("\nNumber of arguments: %d\n", argc); // Print number of arguments passed

for (int i = 0; i < argc; i++) {

printf("\nargv[%d] = %s", i, argv[i]); // Print each argument

}

}

// Function to display employee data

void display\_employee\_data(emp employees[], int count) {

if (count == 0) {

printf("No employee records to display.\n");

return;

}

printf("\nEmployee Records:\n");

printf("%-10s %-20s %-10s %-15s %-10s\n", "ID", "Name", "Gender", "Phone", "Salary");

printf("-------------------------------------------------------------\n");

for (int i = 0; i < count; i++) {

printf("%-10d %-20s %-10c %-15d %-10d\n", employees[i].id, employees[i].Name, employees[i].Gender, employees[i].Phno, employees[i].salary);

}

}

// main.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "employee.h"

int main(int argc, char \*argv[]) {

// Check if the file path is provided as a command-line argument

if (argc != 2) {

printf("Usage: %s <file\_path>\n", argv[0]);

return 1;

}

// Get the file path from the command-line argument

char \*fileName = argv[1];

// Array to store employee records

emp employees[BUFF];

// Display the command-line arguments

display\_arguments(argc, argv);

// Read the employee data from the file

int num\_employees = toreadfromfile(fileName, employees);

// Display the employee data

display\_employee\_data(employees, num\_employees);

return 0;

}