**SDM COLLEGE OF ENGINEERING AND TECHNOLOGY**

Dhavalagiri, Dharwad-580002, Karnataka State, India.

**Email: cse.sdmcet@gmail.com**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**A** [**R**](#Table)**eport**

**on**

**CTA-Assignment**

**COURSE CODE: 22UCSL504 COURSE TITLE: DBMS**

**SEMESTER: 5th DIVISION: A**

**COURSE TEACHER: Dr. U. P. Kulkarni**

**[](#Table)**

**[ Academic Year- 2024-25]**

**Date of Submission: 01-11-2024**

Submitted

By

**Mr. Deepak Kumar P S USN: 2024**

**Table of content**

**1. File Operations……………………………………………………………..……… 3**

**2. Indexing…………………………………………………………………….……… 6**

**3. Accessing excel file……………………………….…………………………….….. 9**

**1. Write the C program to study all file operations related system Calls supported by UNIX OS and C libraries for file operations.**

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <string.h>

void create\_file(const char \*filename) {

int fd = creat(filename, 0644);

if (fd == -1) {

perror("Error creating file");

} else {

printf("File created successfully\n");

close(fd);

}

}

void open\_file(const char \*filename) {

int fd = open(filename, O\_RDONLY);

if (fd == -1) {

perror("Error opening file");

} else {

printf("File opened successfully\n");

close(fd);

}

}

void read\_file(const char \*filename) {

int fd = open(filename, O\_RDONLY);

if (fd == -1) {

perror("Error opening file");

} else {

char buffer[1024];

ssize\_t bytes\_read = read(fd, buffer, sizeof(buffer));

if (bytes\_read == -1) {

perror("Error reading from file");

} else {

printf("Read %zd bytes: %s\n", bytes\_read, buffer);

}

close(fd);

}

}

void write\_file(const char \*filename) {

int fd = open(filename, O\_WRONLY | O\_CREAT, 0644);

if (fd == -1) {

perror("Error opening file");

} else {

char text[1024];

printf("Enter text to write to file: ");

fgets(text, sizeof(text), stdin);

ssize\_t bytes\_written = write(fd, text, strlen(text));

if (bytes\_written == -1) {

perror("Error writing to file");

} else {

printf("Wrote %zd bytes to file\n", bytes\_written);

}

close(fd);

}

}

void delete\_file(const char \*filename) {

if (unlink(filename) == -1) {

perror("Error deleting file");

} else {

printf("File deleted successfully\n");

}

}

int main() {

char filename[1024];

printf("Enter filename: ");

fgets(filename, sizeof(filename), stdin);

filename[strcspn(filename, "\n")] = 0;

int choice;

while (1) {

printf("Enter your choice:\n");

printf("1. Create a file\n");

printf("2. Open a file\n");

printf("3. Read from file\n");

printf("4. Write to file\n");

printf("5. Delete file\n");

printf("6. Quit\n");

scanf("%d", &choice);

getchar();

switch (choice) {

case 1:

create\_file(filename);

break;

case 2:

open\_file(filename);

break;

case 3:

read\_file(filename);

break;

case 4:

write\_file(filename);

break;

case 5:

delete\_file(filename);

break;

case 6:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

**2.Write a C program to demonstrate indexing and associated operations.**

#include <stdio.h>

#define MAX\_SIZE 100

void insert\_element(int arr[], int \*size, int element, int position) {

if (\*size >= MAX\_SIZE) {

printf("Array is full. Cannot insert element.\n");

return;

}

if (position < 0 || position > \*size) {

printf("Invalid position. Please enter a position between 0 and %d.\n", \*size);

return;

}

for (int i = \*size; i > position; i--) {

arr[i] = arr[i - 1];

}

arr[position] = element;

(\*size)++;

printf("Element %d inserted at position %d.\n", element, position);

}

void delete\_element(int arr[], int \*size, int position) {

if (\*size == 0) {

printf("Array is empty. Cannot delete element.\n");

return;

}

if (position < 0 || position >= \*size) {

printf("Invalid position. Please enter a position between 0 and %d.\n", \*size - 1);

return;

}

for (int i = position; i < \*size - 1; i++) {

arr[i] = arr[i + 1];

}

(\*size)--;

printf("Element at position %d deleted.\n", position);

}

void display\_array(int arr[], int size) {

if (size == 0) {

printf("Array is empty.\n");

return;

}

printf("Array elements: ");

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

printf("%d ", arr[i]);

}

printf("\n");

}

int search\_element(int arr[], int size, int element) {

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

if (arr[i] == element) {

return i;

}

}

return -1;

}

int main() {

int arr[MAX\_SIZE];

int size = 0;

int choice, element, position;

while (1) {

printf("\nMenu:\n");

printf("1. Insert Element\n");

printf("2. Delete Element\n");

printf("3. Display Array\n");

printf("4. Search Element\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter element to insert: ");

scanf("%d", &element);

printf("Enter position to insert (0 to %d): ", size);

scanf("%d", &position);

insert\_element(arr, &size, element, position);

break;

case 2:

printf("Enter position to delete (0 to %d): ", size - 1);

scanf("%d", &position);

delete\_element(arr, &size, position);

break;

case 3:

display\_array(arr, size);

break;

case 4:

printf("Enter element to search: ");

scanf("%d", &element);

position = search\_element(arr, size, element);

if (position != -1) {

printf("Element %d found at position %d.\n", element, position);

} else {

printf("Element %d not found in the array.\n", element);

}

break;

case 5:

printf("Exiting...\n");

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

**3. Write the Java program to access the given excel file with known file format.**

import org.apache.poi.ss.usermodel.\*;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

public class ExcelReader {

public static void main(String[] args) {

String excelFilePath = "/Dbms-minor/file.xlsx"; // Update this to the correct path

FileInputStream fileInputStream = null;

Workbook workbook = null;

try {

// Open the Excel file

fileInputStream = new FileInputStream(new File(excelFilePath));

workbook = new XSSFWorkbook(fileInputStream);

// Get the first sheet

Sheet sheet = workbook.getSheetAt(0);

// Iterate through each row in the sheet

for (Row row : sheet) {

// Iterate through each cell in the row

for (Cell cell : row) {

// Print the cell value based on its type

switch (cell.getCellType()) {

case STRING:

System.out.print(cell.getStringCellValue() + "\t");

break;

case NUMERIC:

System.out.print(cell.getNumericCellValue() + "\t");

break;

case BOOLEAN:

System.out.print(cell.getBooleanCellValue() + "\t");

break;

case FORMULA:

System.out.print(cell.getCellFormula() + "\t");

break;

default:

System.out.print("Unknown Cell Type\t");

break;

}

}

System.out.println(); // New line after each row

}

} catch (IOException e) {

e.printStackTrace();

} finally {

// Close resources

try {

if (workbook != null) {

workbook.close();

}

if (fileInputStream != null) {

fileInputStream.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

}