PROGRAMMING IN C

Exam-Oriented Questions & Programming Problems with Solutions

Chapter: User Defined Files

Section A: Exam-Oriented Questions (Short Answer Type)

The following questions are commonly asked in university and semester examinations on this chapter, along with concise, exam-ready answers.

Q1. What is a file? Why is file handling necessary in a C program?

Ans: A file is a named location on a secondary storage device (such as a disk) used to store data permanently. File handling is necessary because data held in ordinary variables exists only in RAM and is lost as soon as the program terminates; storing data in a file allows it to be retained, reused, and shared even after the program has finished running.

Q2. Differentiate between a text file and a binary file.

Ans: A text file stores data as a sequence of human-readable ASCII characters and can be opened directly in any text editor. A binary file stores data exactly as it exists in memory, as raw bytes; it is more compact and preserves precision, but it is not human-readable and requires a program to interpret its contents correctly.

Q3. What is the general syntax of the fopen() function? What does it return if the file cannot be opened?

Ans: The general syntax is: FILE *fp = fopen("filename", "mode"); where filename is the name of the file and mode specifies how it should be opened (e.g. "r", "w", "a"). If the file cannot be opened for the requested mode — for example, opening a non-existent file in "r" mode — fopen() returns NULL, and this should always be checked before using the file pointer.

Q4. List and briefly explain any four file-opening modes used in C.

Ans: "r" opens an existing file for reading only. "w" opens a file for writing, creating a new file or erasing an existing one. "a" opens a file for appending, writing new data at the end without disturbing existing content. "r+" opens an existing file for both reading and writing, without erasing its content.

Q5. Differentiate between fscanf()/fprintf() and fread()/fwrite().

Ans: fscanf() and fprintf() perform formatted (text-based) input and output, converting data to and from human-readable text, similar to scanf() and printf(). fread() and fwrite() perform unformatted (binary) input and output, transferring data exactly as it is represented in memory, without any conversion — making them faster and more precise, but not human-readable.

Q6. What is EOF in the context of file handling? How is it used while reading a file?

Ans: EOF (End Of File) is a special value, usually defined as -1, that certain functions such as fgetc() and fscanf() return when the end of a file has been reached and no more data is available. It is commonly used

as the condition in a loop (for example, while ((ch = fgetc(fp)) != EOF)) to read a file until all its content has been processed.

Q7. What is the purpose of the fclose() function? What may happen if a file is not closed properly?

Ans: fclose() closes an open file, flushing any data still held in the output buffer to the disk and releasing the resources associated with that file pointer. If a file is not closed properly, buffered data written to it may never actually reach the disk, and the operating system may run out of available file handles if many files are left open.

Q8. What is meant by random access in file handling? Which functions are mainly used to achieve it?

Ans: Random access refers to the ability to directly access any record in a file without having to read through all the records that precede it. In C, this is achieved mainly using fseek(), which moves the file pointer to a specific byte position, along with ftell(), which reports the current position, and rewind(), which resets the pointer to the beginning of the file.

Q9. Differentiate between fseek() and rewind().

Ans: fseek(fp, offset, whence) moves the file pointer to any specified byte position within the file, relative to SEEK_SET, SEEK_CUR, or SEEK_END. rewind(fp) always moves the file pointer back to the very beginning of the file; it is effectively a shorthand for fseek(fp, 0, SEEK_SET), with no ability to move to any other position.

Q10. What is the significance of the mode "a+" as compared to "w+"?

Ans: Both "a+" and "w+" open a file for reading as well as writing. However, "w+" always creates a new file or erases the content of an existing one, whereas "a+" preserves the existing content of the file (creating it only if it does not already exist) and ensures that all writes take place at the end of the file, regardless of where the file pointer is positioned for reading.

Section B: Programming Problems with Solutions

The following programming problems are frequently asked in practical examinations and viva-voce sessions. Each problem is accompanied by a complete, working C program and its expected sample output.

Problem 1: Write a C program to create a file and write a few lines of text into it.

Solution:

```
#include <stdio.h>

int main() {
    FILE *fp;
    fp = fopen("myfile.txt", "w");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
    fprintf(fp, "Hello, this is line 1.\n");
    fprintf(fp, "This is line 2.\n");
    fprintf(fp, "File handling in C is easy!\n");
    fclose(fp);
    printf("Data written to file successfully.\n");
    return 0;
}
```

Sample Output:

```
Data written to file successfully.
```

Problem 2: Write a C program to read and display the content of a text file.

Solution:

```
#include <stdio.h>

int main() {
    FILE *fp;
    char ch;
    fp = fopen("myfile.txt", "r");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
    printf("Contents of the file:\n");
    while ((ch = fgetc(fp)) != EOF) {
        putchar(ch);
    }
    fclose(fp);
    return 0;
}
```

Sample Output:

```
Contents of the file:
Hello, this is line 1.
This is line 2.
File handling in C is easy!
```

Problem 3: Write a C program to copy the contents of one file into another file.

Solution:

```
#include <stdio.h>

int main() {
    FILE *src, *dest;
    char ch;

    src = fopen("myfile.txt", "r");
    if (src == NULL) {
        printf("Cannot open source file!\n");
        return 1;
    }

    dest = fopen("copy.txt", "w");
    if (dest == NULL) {
        printf("Cannot open destination file!\n");
        fclose(src);
        return 1;
    }

    while ((ch = fgetc(src)) != EOF) {
        fputc(ch, dest);
    }

    printf("File copied successfully.\n");
    fclose(src);
    fclose(dest);
    return 0;
}
```

Sample Output:

```
File copied successfully.
```

Problem 4: Write a C program to count the number of characters, words, and lines in a text file.

Solution:

```
#include <stdio.h>

int main() {
    FILE *fp;
    char ch;
    int chars = 0, words = 0, lines = 0;

    fp = fopen("myfile.txt", "r");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    while ((ch = fgetc(fp)) != EOF) {
        chars++;
        if (ch == ' ' || ch == '\n')
```

```
            words++;
        if (ch == '\n')
            lines++;
    }

    fclose(fp);
    printf("Characters: %d\n", chars);
    printf("Words: %d\n", words);
    printf("Lines: %d\n", lines);
    return 0;
}
```

Sample Output:

```
Characters: 66
Words: 12
Lines: 3
```

Problem 5: Write a C program to append new data at the end of an existing file.

Solution:

```
#include <stdio.h>

int main() {
    FILE *fp;
    fp = fopen("myfile.txt", "a");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
    fprintf(fp, "This line is appended.\n");
    fclose(fp);
    printf("Data appended successfully.\n");
    return 0;
}
```

Sample Output:

```
Data appended successfully.
```

Problem 6: Write a C program to store student records (using a structure) in a binary file and then read them back.

Solution:

```
#include <stdio.h>

struct Student {
    int roll;
    char name[30];
    float marks;
};

int main() {
    struct Student s[3] = {
        {101, "Ravi", 78.5},
        {102, "Sita", 88.0},
```

```
        {103, "Aman", 65.25}
    };
    struct Student temp;
    FILE *fp;

    fp = fopen("students.dat", "wb");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
    fwrite(s, sizeof(struct Student), 3, fp);
    fclose(fp);

    fp = fopen("students.dat", "rb");
    printf("Roll\tName\tMarks\n");
    while (fread(&temp, sizeof(struct Student), 1, fp) == 1) {
        printf("%d\t%s\t%.2f\n", temp.roll, temp.name, temp.marks);
    }
    fclose(fp);
    return 0;
}
```

Sample Output:

```
Roll     Name  Marks
101Ravi  78.50
102Sita  88.00
103Aman  65.25
```

Problem 7: Write a C program to access and display a particular (nth) record from a binary file using fseek().

Solution:

```
#include <stdio.h>

struct Student {
    int roll;
    char name[30];
    float marks;
};

int main() {
    FILE *fp;
    struct Student s;
    int n;

    fp = fopen("students.dat", "rb");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    printf("Enter record number to view: ");
    scanf("%d", &n);
```

```
    fseek(fp, (n - 1) * sizeof(struct Student), SEEK_SET);
    fread(&s, sizeof(struct Student), 1, fp);

    printf("Roll: %d\nName: %s\nMarks: %.2f\n", s.roll, s.name, s.marks);

    fclose(fp);
    return 0;
}
```

Sample Output:

```
Enter record number to view: 2
Roll: 102
Name: Sita
Marks: 88.00
```

Problem 8: Write a C program to count the total number of records stored in a binary file using fseek() and ftell().

Solution:

```
#include <stdio.h>

struct Student {
    int roll;
    char name[30];
    float marks;
};

int main() {
    FILE *fp;
    long size;
    int count;

    fp = fopen("students.dat", "rb");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    fseek(fp, 0, SEEK_END);
    size = ftell(fp);
    count = size / sizeof(struct Student);

    printf("Total number of records: %d\n", count);

    fclose(fp);
    return 0;
}
```

Sample Output:

```
Total number of records: 3
```

Problem 9: Write a C program to merge the contents of two files into a third file.

Solution:

```c
#include <stdio.h>

int main() {
    FILE *f1, *f2, *f3;
    char ch;

    f1 = fopen("file1.txt", "r");
    f2 = fopen("file2.txt", "r");
    f3 = fopen("merged.txt", "w");

    if (f1 == NULL || f2 == NULL || f3 == NULL) {
        printf("Error opening one of the files!\n");
        return 1;
    }

    while ((ch = fgetc(f1)) != EOF)
        fputc(ch, f3);

    while ((ch = fgetc(f2)) != EOF)
        fputc(ch, f3);

    printf("Files merged successfully into merged.txt\n");

    fclose(f1);
    fclose(f2);
    fclose(f3);
    return 0;
}
```

Sample Output:

```
Files merged successfully into merged.txt
```

Problem 10: Write a C program to check whether a given file exists or not.

Solution:

```c
#include <stdio.h>

int main() {
    FILE *fp;
    char filename[50];

    printf("Enter file name to check: ");
    scanf("%s", filename);

    fp = fopen(filename, "r");
    if (fp == NULL) {
        printf("File does not exist.\n");
    } else {
        printf("File exists.\n");
        fclose(fp);
    }
    return 0;
}
```

Sample Output:

```
Enter file name to check: myfile.txt
File exists.
```