QUESTION 1

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    FILE *fp1, *fp2, *fp3;
    char ch;

    fp1 = fopen("file1.txt", "w");
    fprintf(fp1, "This is the content of file1.");
    fclose(fp1);

    fp2 = fopen("file2.txt", "w");
    fprintf(fp2, "This is the content of file2.");
    fclose(fp2);

    fp1 = fopen("file1.txt", "r");
    fp2 = fopen("file2.txt", "r");
    fp3 = fopen("file3.txt", "w");

    if (fp1 == NULL || fp2 == NULL || fp3 == NULL) {
        printf("Error opening file.\n");
        exit(1);
    }

    while ((ch = fgetc(fp1)) != EOF) {
        fputc(ch, fp3);
    }


    while ((ch = fgetc(fp2)) != EOF) {
        fputc(ch, fp3);
    }

    printf("File merged successfully.\n");

    fclose(fp1);
    fclose(fp2);
    fclose(fp3);

    return 0;
}
```

QUESTION 2

```
#include <stdio.h>
#include <ctype.h>

#define MAX_SIZE 1000

int main() {
    char file_name[100], c;
    int freq[26] = {0};
    FILE *fp, *fptr;

    printf("Enter the name of file to read from: ");
    scanf("%s", file_name);

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

    while ((c = fgetc(fp)) != EOF) {
        if (isalpha(c)) {
            freq[toupper(c) - 'A']++;
        }
    }

    fclose(fp);

    fptr = fopen("output.txt", "w");
    if (fptr == NULL) {
        printf("Error opening file!\n");
        return 1;
    }

    int total_chars = 0;
    for (int i = 0; i < 26; i++) {
        if (freq[i] > 0) {
            fprintf(fptr, "%c (%d)\n", 'A' + i, freq[i]);
            total_chars += freq[i];
        }
    }

    fprintf(fptr, "Total characters read = %d", total_chars);
    fclose(fptr);

    return 0;
}
```

QUESTION 3

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SIZE 100

typedef struct {
    int id;
    char name[MAX_SIZE];
    char sex[MAX_SIZE];
    int quiz1, quiz2;
    int midterm;
    int final;
    int total;
} Student;

int main() {
    FILE *fp;
    char file_name[MAX_SIZE], option[MAX_SIZE];
    int num_students = 0;
    Student students[20];
    int num_quizzes = 2;
    int num_students_to_process = 0;
    int score_cutoffs[3] = {50, 80, 100};
    int num_scores[3][7] = {0};

    printf("Enter the name of the file to store records: ");
    scanf("%s", file_name);

    fp = fopen(file_name, "w");
    if (fp == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
```

```
printf("Enter student records (ID, Name, Sex, Quiz 1, Quiz 2, Midterm, Final):\n");

while (num_students < 20) {
   Student student;
   scanf("%d %s %s %d %d %d %d", &student.id, student.name, student.sex,
        &student.quiz1, &student.quiz2, &student.midterm, &student.final);

   student.total = student.quiz1 + student.quiz2 + student.midterm + student.final;

   students[num_students++] = student;

   fprintf(fp, "%d %s %s %d %d %d %d %d\n", student.id, student.name, student.sex,
         student.quiz1, student.quiz2, student.midterm, student.final, student.total);

   printf("Enter another record (Y/N)? ");
   scanf("%s", option);

   if (strcmp(option, "N") == 0) {
      break;
   }
}

fclose(fp);

printf("\n");

printf("Enter the number of students to process (1-20): ");
scanf("%d", &num_students_to_process);

if (num_students_to_process < 1 || num_students_to_process > 20) {
   printf("Invalid number of students!\n");
   return 1;
}

printf("\n");

printf("ID\tName\t\tSex\tQuiz 1\tQuiz 2\tMidterm\tFinal\tTotal\n");

for (int i = 0; i < num_students_to_process; i++) {
   Student student = students[i];

   printf("%d\t%s\t%s\t%d\t%d\t%d\t%d\t%d\n", student.id, student.name, student.sex,
        student.quiz1, student.quiz2, student.midterm, student.final, student.total);
        for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 7; k++) {
                if (student.total < score_cutoffs[j] && student.total >= score_cutoffs[j-1]) {
                    num_scores[j][k]++;
                    break;
                }
            }
        }
    }

    printf("\n");

    printf("Score Ranges:\n");
    printf("< 50\t");
    for (int i = 0; i < num_scores[0][0]; i++) {
        printf("*");
    }
    printf("\n");

    printf("50-79\t");
    for (int i = 0; i < num_scores[1][0]; i++) {
printf("*");
}
printf("\n");
printf("80-99\t");
for (int i = 0; i < num_scores[2][0]; i++) {
    printf("*");
}
printf("\n");

printf("100\t");
for (int i = 0; i < num_scores[2][1]; i++) {
    printf("*");
}
printf("\n");

printf("> 100\t");
for (int i = 0; i < num_scores[2][2]; i++) {
    printf("*");
}
printf("\n");

printf("\n");
```

```
return 0;

}
```

QUESTION 4

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_TOOLS 10
#define TOOL_NAME_LENGTH 20
#define FILE_NAME "hardware.txt"

typedef struct {
    char name[TOOL_NAME_LENGTH];
    int quantity;
    float cost;
} tool_t;

int read_tool(FILE *fp, int id, tool_t *tool) {
    if (fseek(fp, id * sizeof(tool_t), SEEK_SET) != 0) {
        perror("fseek");
        return -1;
    }
    if (fread(tool, sizeof(tool_t), 1, fp) != 1) {
        perror("fread");
        return -1;
    }
    return 0;
}

int write_tool(FILE *fp, int id, const tool_t *tool) {
    if (fseek(fp, id * sizeof(tool_t), SEEK_SET) != 0) {
        perror("fseek");
        return -1;
    }
    if (fwrite(tool, sizeof(tool_t), 1, fp) != 1) {
        perror("fwrite");
        return -1;
    }
    return 0;
}
```

```
void print_tool(const tool_t *tool, int id) {
    printf("%2d %-20s %3d %6.2f\n", id, tool->name, tool->quantity, tool->cost);
}

int main() {
    FILE *fp;
    tool_t tool;
    int id, choice;

    fp = fopen(FILE_NAME, "r+");
    if (fp == NULL) {
        perror("fopen");
        exit(EXIT_FAILURE);
    }

    if (fseek(fp, 0, SEEK_END) != 0) {
        perror("fseek");
        exit(EXIT_FAILURE);
    }

    if (ftell(fp) < MAX_TOOLS * sizeof(tool_t)) {
        memset(&tool, 0, sizeof(tool_t));
        for (int i = 0; i < MAX_TOOLS; i++) {
            if (fwrite(&tool, sizeof(tool_t), 1, fp) != 1) {
                perror("fwrite");
                exit(EXIT_FAILURE);
            }
        }
    }

    do {
        printf("\nMenu\n");
        printf("1. List all tools\n");
        printf("2. Add or update tool\n");
        printf("3. Delete tool\n");
        printf("4. Quit\n");
        printf("Enter choice (1-4): ");
        if (scanf("%d", &choice) != 1) {
            printf("Invalid choice\n");
            continue;
        }
        switch (choice) {
            case 1:
                printf("\nTool list:\n");
                printf("ID  Name                Qty   Cost\n");
                for (int i = 0; i < MAX_TOOLS; i++) {
                    if (read_tool(fp, i, &tool) == 0) {
                        if (tool.quantity > 0) {
                            print_tool(&tool, i);
                        }
                    }
                }
                break;
            case 2:
                printf("\nEnter tool ID (0-%d): ", MAX_TOOLS - 1);
                if (scanf("%d", &id) != 1 || id < 0 || id >= MAX_TOOLS) {
                    printf("Invalid ID\n");
                    continue;
                }
                if (read_tool(fp, id, &tool) != 0) {
                    printf("Failed to read tool\n");
                    continue;
                }
                printf("Enter tool name (max %d characters): ", TOOL_NAME_LENGTH - 1);
                if (scanf("%s", tool.name) !=1 || strlen(tool.name) >= TOOL_NAME_LENGTH) {
printf("Invalid name\n");
continue;
}
printf("Enter quantity: ");
if (scanf("%d", &tool.quantity) != 1 || tool.quantity < 0) {
printf("Invalid quantity\n");
continue;
}
printf("Enter cost: ");
if (scanf("%f", &tool.cost) != 1 || tool.cost < 0) {
printf("Invalid cost\n");
continue;
}
if (write_tool(fp, id, &tool) != 0) {
printf("Failed to write tool\n");
continue;
}
printf("Tool updated\n");
break;
case 3:
printf("\nEnter tool ID (0-%d): ", MAX_TOOLS - 1);
if (scanf("%d", &id) != 1 || id < 0 || id >= MAX_TOOLS) {
```

```
printf("Invalid ID\n");
continue;
}
if (read_tool(fp, id, &tool) != 0) {
printf("Failed to read tool\n");
continue;
}
if (tool.quantity == 0) {
printf("This tool is not in stock\n");
continue;
}
memset(&tool, 0, sizeof(tool_t));
if (write_tool(fp, id, &tool) != 0) {
printf("Failed to delete tool\n");
continue;
}
printf("Tool deleted\n");
break;
case 4:
break;
default:
printf("Invalid choice\n");
}
} while (choice != 4);

fclose(fp);

return 0;
}
```

QUESTION 5

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_STUDENTS 100

typedef struct Student {
    int RollNo;
    char Name[50];
    char Department[50];
    int Batch;
    char Section[5];
    float CGPA;
```

```
} Student;

void write_students(Student students[], int num_students, char* filename) {
    FILE* fp;
    fp = fopen(filename, "w");

    if (fp == NULL) {
        printf("Error opening file %s\n", filename);
        exit(1);
    }

    for (int i = 0; i < num_students; i++) {
        fprintf(fp, "%d,%s,%s,%d,%s,%.2f\n", students[i].RollNo, students[i].Name,
students[i].Department, students[i].Batch, students[i].Section, students[i].CGPA);
    }

    fclose(fp);
}

void read_students(Student students[], int* num_students, char* filename) {
    FILE* fp;
    fp = fopen(filename, "r");

    if (fp == NULL) {
        printf("Error opening file %s\n", filename);
        exit(1);
    }

    char line[256];
    int i = 0;
    while (fgets(line, sizeof(line), fp)) {
        sscanf(line, "%d,%[^,],%[^,],%d,%[^,],%f", &students[i].RollNo, students[i].Name,
students[i].Department, &students[i].Batch, students[i].Section, &students[i].CGPA);
        i++;
    }

    *num_students = i;
    fclose(fp);
}

void search_student_by_rollNo(Student students[], int num_students, int rollNo) {
    for (int i = 0; i < num_students; i++) {
        if (students[i].RollNo == rollNo) {
            printf("%d,%s,%s,%d,%s,%.2f\n", students[i].RollNo, students[i].Name,
students[i].Department, students[i].Batch, students[i].Section, students[i].CGPA);
            return;
        }
    }
    printf("No student with RollNo %d found.\n", rollNo);
}

void print_students_in_batch(Student students[], int num_students, int batch) {
    for (int i = 0; i < num_students; i++) {
        if (students[i].Batch == batch) {
            printf("%d,%s,%s,%d,%s,%.2f\n", students[i].RollNo, students[i].Name,
students[i].Department, students[i].Batch, students[i].Section, students[i].CGPA);
        }
    }
}

int main() {
    Student students[MAX_STUDENTS];
    int num_students = 0;


    students[num_students++] = (Student){ 101, "John", "CS", 2022, "A", 3.8 };
    students[num_students++] = (Student){ 102, "Jane", "EE", 2021, "B", 3.5 };
    students[num_students++] = (Student){ 103, "Alice", "ME", 2022, "C", 3.2 };
    students[num_students++] = (Student){ 104, "Bob", "CE", 2023,"D", 3.9 };

write_students(students, num_students, "students.txt");
read_students(students, &num_students, "students.txt");


int rollNo;
printf("Enter RollNo to search: ");
scanf("%d", &rollNo);
search_student_by_rollNo(students, num_students, rollNo);


printf("Students in Batch 2022:\n");
print_students_in_batch(students, num_students, 2022);

return 0;
}
```

QUESTION 6

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

int is_vowel(char c) {
    char vowels[] = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'};
    int i;
    for (i = 0; i < 10; i++) {
        if (c == vowels[i]) {
            return 1;
        }
    }
    return 0;
}

int main() {
    FILE *fp_in, *fp_out;
    char filename_in[50], filename_out[50];
    printf("Enter the name of the input file: ");
    scanf("%s", filename_in);
    printf("Enter the name of the output file: ");
    scanf("%s", filename_out);
    fp_in = fopen(filename_in, "r");
    if (fp_in == NULL) {
        printf("Error: can't open file %s.\n", filename_in);
        exit(1);
    }
    fp_out = fopen(filename_out, "w");
    if (fp_out == NULL) {
        printf("Error: can't create file %s.\n", filename_out);
        exit(1);
    }
    int c, count = 0, s_count = 0;
    while ((c = fgetc(fp_in)) != EOF) {
        if (is_vowel(c)) {
            count++;
            if (count % 2 == 1) {
                if (islower(c)) {
                    fprintf(fp_out, "vow");
                } else {
                    fprintf(fp_out, "VOW");
                }
            } else {
                fprintf(fp_out, "%c", c);
            }
        } else if (c == 's' || c == 'S') {
            s_count++;
            if (s_count == 3) {
                fprintf(fp_out, "PF-Lab");
                s_count = 0;
            }
        } else {
            fprintf(fp_out, "%c", c + 3);
        }
    }
    fclose(fp_in);
    fclose(fp_out);
    printf("File encryption completed.\n");
    return 0;
}
```