1. Write a program to create and write to a text file.
IPO
input :enter a value as a input
process:to create and write to a text file.
Output:output the variable

Program

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

int main(void) {
    const char *filename = "output.txt";
    const char *text = "Hello, this is a sample text written to the file using C.\n";

      FILE *file = fopen(filename, "w");
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }

       if (fputs(text, file) == EOF) {
        perror("Error writing to file");
        fclose(file);
        return EXIT_FAILURE;
    }

       if (fclose(file) != 0) {
        perror("Error closing file");
        return EXIT_FAILURE;
    }

    printf("Successfully wrote to '%s'\n", filename);
    return EXIT_SUCCESS;
}
```

Output

Output

```
Error opening file for writing: Permission denied
```

2. Write a program to read contents of a file and display.
IPO
input :enter a value as a input
process:to create and write to a text file.
Output:output the variable

Program

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

int main(void) {
    const char* filename = "input.txt";
    FILE* file = fopen(filename, "r");
    if (file == NULL) {
        perror("Error opening file for reading");
        return EXIT_FAILURE;
    }

    printf("Contents of '%s':\n", filename);
    int ch;
    while ((ch = fgetc(file)) != EOF) {
        putchar(ch);
    }


    if (fclose(file) != 0) {
        perror("Error closing file");
        return EXIT_FAILURE;
    }

    return EXIT_SUCCESS;
}
```

Output

Output

```
Error opening file for reading: No such file or directory
```

3. Write a program to count number of lines in a file.

IPO
input :enter a value as a input
process:to count number of lines in a file.
Output:output the variable

Program

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

int main(void) {
    const char *filename = "input.txt";
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }

    int ch;
    unsigned long line_count = 0;
    while ((ch = fgetc(file)) != EOF) {
        if (ch == '\n') {
            ++line_count;
        }
    }
    if (line_count == 0 && ftell(file) > 0) {
        line_count = 1;
    } else if (ftell(file) > 0) {

        fseek(file, -1, SEEK_END);
        if (fgetc(file) != '\n') {
            ++line_count;
        }
    }

    fclose(file);

    printf("The file '%s' has %lu line%s.\n", filename, line_count,
           line_count == 1 ? "" : "s");

    return EXIT_SUCCESS;
}
```

Output

Output

```
Error opening file: No such file or directory
```

4. Write a program to copy contents from one file to another.
IPO
input :enter a value as a input
process:to copy contents from one file to another.
Output:output the variable

Program

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

int main(void) {
    const char *sourceFile = "source.txt";
    const char *destFile   = "destination.txt";


    FILE *src = fopen(sourceFile, "r");
    if (src == NULL) {
        perror("Error opening source file");
        return EXIT_FAILURE;
    }


    FILE *dst = fopen(destFile, "w");
    if (dst == NULL) {
        perror("Error opening destination file");
        fclose(src);
        return EXIT_FAILURE;
    }


    char buffer[4096];
    size_t bytesRead;


    while ((bytesRead = fread(buffer, 1, sizeof(buffer), src)) > 0) {
        size_t bytesWritten = fwrite(buffer, 1, bytesRead, dst);
        if (bytesWritten != bytesRead) {
            perror("Error writing to destination file");
            fclose(src);
            fclose(dst);
```

```
        return EXIT_FAILURE;
    }
  }

  if (ferror(src)) {
    perror("Error reading from source file");
  }


  fclose(src);
  fclose(dst);

  printf("Contents copied from '%s' to '%s' successfully.\n", sourceFile, destFile);
  return EXIT_SUCCESS;
}
```

Output

Output

```
Error opening source file: No such file or directory
```

5. Write a program to append text to a file.

IPO

input :enter a value as a input

process:to append text to a file.

Output:output the variable

Program

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

int main(void) {
  const char *filename = "example.txt";
  const char *textToAppend = "This is the text being appended.\n";


  FILE *file = fopen(filename, "a");
  if (file == NULL) {
    perror("Error opening file for appending");
    return EXIT_FAILURE;
  }

    if (fputs(textToAppend, file) == EOF) {
```

```
        perror("Error writing to file");
        fclose(file);
        return EXIT_FAILURE;
    }

    fclose(file);
    printf("Successfully appended to '%s'.\n", filename);

      file = fopen(filename, "r");
    if (file == NULL) {
        perror("Error opening file for reading");
        return EXIT_FAILURE;
    }

    printf("\nContents of '%s' after appending:\n", filename);
    int ch;
    while ((ch = fgetc(file)) != EOF) {
        putchar(ch);
    }

    fclose(file);
    return EXIT_SUCCESS;
}
```

Output
Successfully appended to 'example.txt'.

Contents of 'example.txt' after appending:
Hello, world!
This is the initial content.
This is the text being appended.

6. Write a program to count vowels in a file.
IPO
input :enter a value as a input
process:to count vowels in a file.
Output:output the variable

Program

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

```
int is_vowel(char ch) {
    ch = tolower((unsigned char)ch);
    return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u');
}

int main(void) {
    const char *filename = "input.txt";
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }

    unsigned long vowel_count = 0;
    unsigned long total_count = 0;
    int ch;

    while ((ch = fgetc(file)) != EOF) {
        total_count++;
        if (is_vowel(ch)) {
            vowel_count++;
        }
    }

    if (ferror(file)) {
        perror("Error reading the file");
        fclose(file);
        return EXIT_FAILURE;
    }

    fclose(file);

    printf("File: '%s'\n", filename);
    printf("Total characters read: %lu\n", total_count);
    printf("Number of vowels: %lu\n", vowel_count);

    return EXIT_SUCCESS;
}
```

Output
File: 'input.txt'
Total characters read: 29
Number of vowels: 10

7. Write a program to read integers from a file and find the sum.
IPO
input :enter a value as a input
process:to read integers from a file and find the sum.
Output:output the variable

Program
Program

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

int main(void) {
    const char *filename = "numbers.txt";
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }

    long sum = 0;
    int value;
    unsigned long count = 0;

    // Read integers until EOF
    while (fscanf(file, "%d", &value) == 1) {
        sum += value;
        count++;
    }

    if (ferror(file)) {
        perror("Error reading from file");
        fclose(file);
        return EXIT_FAILURE;
    }

    fclose(file);

    printf("Read %lu integer%s from '%s'.\n", count, (count == 1 ? "" : "s"), filename);
    printf("Sum of the integers: %ld\n", sum);

    return EXIT_SUCCESS;
}
```

Output
10
-3
24
4
Read 4 integers from 'numbers.txt'.
Sum of the integers: 36

8. Write a program to read a structure from a file.
IPO
input :enter a value as a input
process:to read a structure from a file.
Output:output the variable

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

typedef struct {
    int id;
    char name[50];
    float salary;
} Employee;

int main(void) {
    const char *filename = "employees.dat";
    FILE *file = fopen(filename, "rb");  // Open in binary read mode
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }

    Employee emp;

    // Read the structure from the file
    if (fread(&emp, sizeof(Employee), 1, file) != 1) {
        if (feof(file)) {
            fprintf(stderr, "Unexpected end of file; no structure to read.\n");
        } else {
            perror("Error reading structure from file");
        }
        fclose(file);
        return EXIT_FAILURE;
```

```
    }

    fclose(file);

       printf("Employee Details (read from file):\n");
    printf("ID     : %d\n", emp.id);
    printf("Name   : %s\n", emp.name);
    printf("Salary : %.2f\n", emp.salary);

    return EXIT_SUCCESS;
}
```

Output
Employee Details (read from file):
ID   : 101
Name   : Alice Johnson
Salary : 75000.00

9. Write a program to sort names stored in a file.
IPO
input :enter a value as a input
process:to sort names stored in a file.
Output:output the variable

Program

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

#define MAX_NAMES 1000
#define MAX_LEN   100

int main(void) {
    const char *inputFile = "names.txt";
    const char *outputFile = "sorted_names.txt";
    char *names[MAX_NAMES];
    int count = 0;

    FILE *fin = fopen(inputFile, "r");
    if (fin == NULL) {
        perror("Error opening input file");
        return EXIT_FAILURE;
    }
```

```
    char buffer[MAX_LEN];
    while (fgets(buffer, sizeof(buffer), fin) != NULL) {
        buffer[strcspn(buffer, "\r\n")] = '\0';
        names[count] = malloc(strlen(buffer) + 1);
        if (names[count] == NULL) {
            perror("Memory allocation failed");
            return EXIT_FAILURE;
        }
        strcpy(names[count], buffer);
        count++;
        if (count >= MAX_NAMES) break;
    }
    fclose(fin);

        for (int i = 0; i < count - 1; i++) {
        for (int j = i + 1; j < count; j++) {
            if (strcmp(names[i], names[j]) > 0) {
                char *tmp = names[i];
                names[i] = names[j];
                names[j] = tmp;
            }
        }
    }

    FILE *fout = fopen(outputFile, "w");
    if (fout == NULL) {
        perror("Error opening output file");
        return EXIT_FAILURE;
    }

    for (int i = 0; i < count; i++) {
        fprintf(fout, "%s\n", names[i]);
        free(names[i]);
    }
    fclose(fout);

    printf("Sorted %d names and saved to '%s'.\n", count, outputFile);
    return EXIT_SUCCESS;
}
```

Output
Alice
Bob

Charlie
Diana

10. Write a program to search for a word in a file.
IPO
input :enter a value as a input
process:to search for a word in a file.
Output:output the variable

program

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

#define MAX_LINE_LEN 1024

int main(void) {
    const char *filename = "input.txt";
    const char *search_word = "target";  /
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }

    char line[MAX_LINE_LEN];
    unsigned long line_number = 0;
    unsigned long match_count = 0;

    while (fgets(line, sizeof(line), file) != NULL) {
        line_number++;
        if (strstr(line, search_word) != NULL) {
            printf("Found \"%s\" on line %lu: %s", search_word, line_number, line);
            match_count++;
        }
    }

    if (ferror(file)) {
        perror("Error reading file");
        fclose(file);
        return EXIT_FAILURE;
    }
```

```
    fclose(file);

    if (match_count == 0) {
        printf("\"%s\" not found in \"%s\".\n", search_word, filename);
    } else {
        printf("\nTotal occurrences found: %lu\n", match_count);
    }

    return EXIT_SUCCESS;
}
```

Output
Found "target" on line 3: We are searching for a target phrase in this text.
Found "target" on line 4: Another line with the target word: target.

Total occurrences found: 2