NAME - SHREERAM
Reg No. - 20214033

ASSIGNMENT - 2

1)

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

#define MAX_LENGTH 100

void cdCommand(char *path) {
    if (chdir(path) == -1) {
        perror("cd");
    }
}

void pwdCommand() {
    char cwd[MAX_LENGTH];
    if (getcwd(cwd, sizeof(cwd)) != NULL) {
        printf("%s\n", cwd);
    } else {
        perror("pwd");
    }
}

void mkdirCommand(char *dirName) {
    if (mkdir(dirName, 0777) == -1) {
        perror("mkdir");
    }
}

void rmdirCommand(char *dirName) {
    if (rmdir(dirName) == -1) {
```

```
        perror("rmdir");
    }
}

void lsCommand() {
    struct dirent *entry;
    DIR *dir = opendir(".");
    while ((entry = readdir(dir)) != NULL) {
        printf("%s\n", entry->d_name);
    }

    closedir(dir);
}

int main() {
    char input[MAX_LENGTH];
    char command[MAX_LENGTH];
    char argument[MAX_LENGTH];

    while (1) {
        printf("myShell$ ");
        fgets(input, MAX_LENGTH, stdin);

        sscanf(input, "%s %s", command, argument);

        if (strcmp(command, "cd") == 0) {
            cdCommand(argument);
        } else if (strcmp(command, "pwd") == 0) {
            pwdCommand();
        } else if (strcmp(command, "mkdir") == 0) {
            mkdirCommand(argument);
        } else if (strcmp(command, "rmdir") == 0) {
            rmdirCommand(argument);
        } else if (strcmp(command, "ls") == 0) {
            lsCommand();
```

```
        } else if (strcmp(command, "exit") == 0) {
            exit(0);
        } else {
            printf("Invalid command\n");
        }
    }

    return 0;
}
```

2)

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

int searchFile(const char *dirPath, const char *fileName) {
    DIR *dir = opendir(dirPath);

    struct dirent *entry;
    while ((entry = readdir(dir)) != NULL) {
        if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") ==
0) {
            continue;
        }

        if (strcmp(entry->d_name, fileName) == 0) {
            printf("Found: %s/%s\n", dirPath, fileName);
            return 1;
        }

        if (entry->d_type == DT_DIR) {
            char newPath[1024];
```

```
            snprintf(newPath, sizeof(newPath), "%s/%s", dirPath,
entry->d_name);
            if (searchFile(newPath, fileName)) {
                closedir(dir);
                return 1;
            }
        }
    }

    closedir(dir);
    return 0;
}

int main(int argc, char *argv[]) {
    const char *startingDir = argv[1];
    const char *fileName = argv[2];

    if (searchFile(startingDir, fileName)) {
        printf("File found!\n");
    } else {
        printf("File not found.\n");
    }

    return 0;
}
```

3)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <sys/stat.h>
#include <unistd.h>
```

```
void deleteDirectory(const char *dirPath) {
    DIR *dir = opendir(dirPath);

    struct dirent *entry;
    while ((entry = readdir(dir)) != NULL) {
        if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") ==
0) {
            continue;
        }

        char newPath[1024];
        snprintf(newPath, sizeof(newPath), "%s/%s", dirPath, entry->d_name);

        struct stat statbuf;

        if (S_ISDIR(statbuf.st_mode)) {
            deleteDirectory(newPath);
        } else {
            unlink(newPath);
        }
    }

    closedir(dir);
    rmdir(dirPath);
}

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

    const char *dirPath = argv[1];

    printf("Directory '%s' and its subfolders deleted.\n", dirPath);

    return 0;
}
```

4)

```
#!/bin/bash

while true; do
    echo "Menu:"
    echo "1. Merge contents of two files"
    echo "2. Search for a pattern in a file"
    echo "3. Exit"
    read -p "Enter your choice: " choice

    case $choice in
        1)
            read -p "Enter the first file name: " file1
            read -p "Enter the second file name: " file2
            read -p "Enter the output file name: " outputFile

            cat "$file1" "$file2" > "$outputFile"
            echo "Contents of $file1 and $file2 have been merged into
$outputFile"
            ;;
        2)
            read -p "Enter the file name: " file
            read -p "Enter the pattern to search: " pattern

            if grep -q "$pattern" "$file"; then
                echo "Pattern '$pattern' found in $file"
            else
                echo "Pattern '$pattern' not found in $file"
            fi
            ;;
        3)
            echo "Exiting..."
            exit 0
            ;;
```

```
        *)
            echo "Invalid option"
            ;;
    esac

    echo
done



5)
#!/bin/bash

while true; do
    echo "Menu:"
    echo "1. Number of active users"
    echo "2. Display lines from the top of a file"
    echo "3. Update access time of a file"
    echo "4. Exit"
    read -p "Enter your choice: " choice

    case $choice in
        1)
            activeUsers=$(who | wc -l)
            echo "Number of active users: $activeUsers"
            ;;
        2)
            read -p "Enter the file name: " file
            read -p "Enter the number of lines to display: " numLines

            if [[ -f "$file" ]]; then
                echo "Top $numLines lines of $file:"
                head -n "$numLines" "$file"
            else
                echo "$file doesn't exist."
            fi
```

```
        ;;
    3)
        read -p "Enter the file name: " file

        if [[ -f "$file" ]]; then
            touch "$file"
            echo "Access time of $file updated to current time."
        else
            echo "$file doesn't exist."
        fi
        ;;
    4)
        echo "Exiting..."
        exit 0
        ;;
    *)
        echo "Invalid option"
        ;;
  esac

  echo
done
```