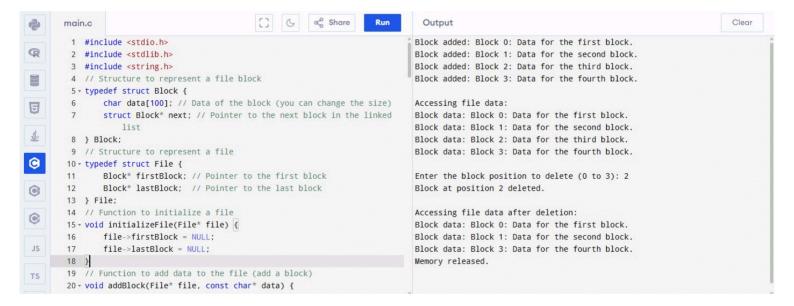
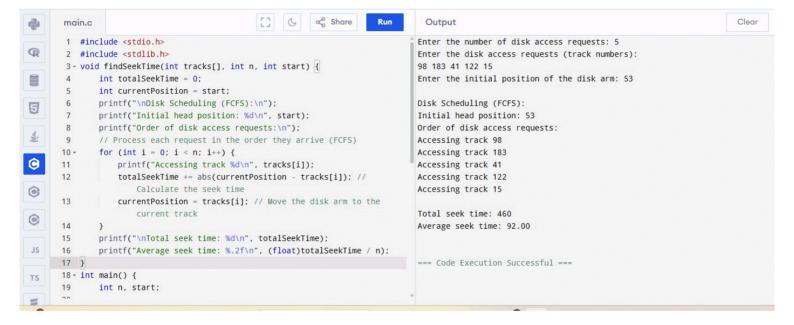
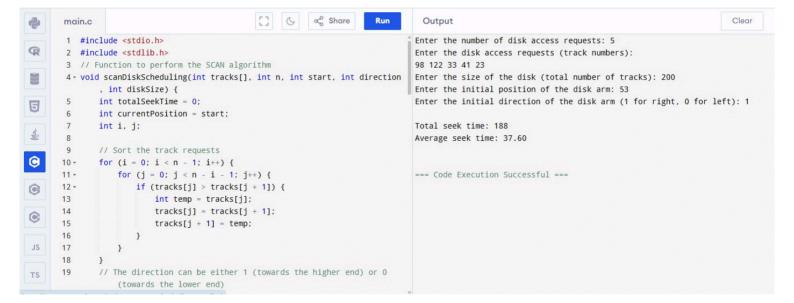
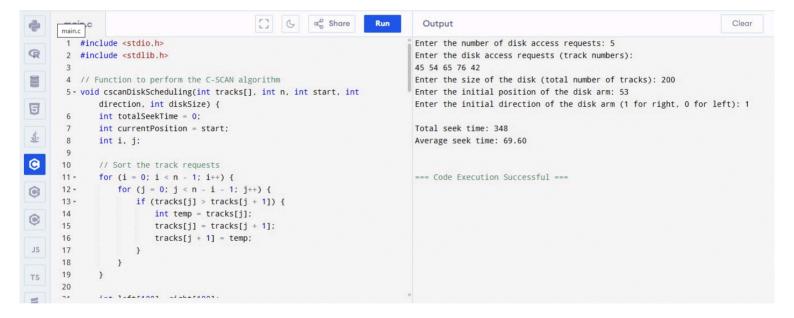
```
®<sub>⊚</sub> Execute
                                                                   Enter record index to access (0 to 4) or -1 to exit: 2
                                                                   Accessing record 2: Record 3: Data for third record
   4 #define MAX_RECORDS 10
                                                                   Enter record index to access (0 to 4) or -1 to exit: 4
     #define MAX_RECORD_SIZE 100
                                                                   Accessing record 4: Record 5: Data for fifth record
                                                                   Enter record index to access (0 to 4) or -1 to exit: -1
   8 - typedef struct {
         char records[MAX_RECORDS][MAX_RECORD_SIZE];
         int totalRecords;
 10
 11 } File;
     void initializeFile(File* file) {
         file->totalRecords = 0;
     }
 17 - void addRecord(File* file, const char* record) {
         if (file->totalRecords < MAX_RECORDS) {</pre>
 19
             strcpy(file->records[file->totalRecords], record);
 20
             file->totalRecords++;
         } else {
 21
             printf("File is full. Cannot add more records.\n"
```

```
∞ Share
                                                                                  Output
                                                                                                                                                    Clear
       main.c
       1 #include <stdio.h>
                                                                                 Data added to block 0: Block 0: Data for the first block.
                                                                                 Data added to block 1: Block 1: Data for the second block.
       2 #include <stdlib.h>
       3 #include <string.h>
                                                                                 Data added to block 2: Block 2: Data for the third block.
       4 #define MAX_BLOCKS 10 // Maximum number of blocks for a file
                                                                                 Data added to block 3: Block 3: Data for the fourth block.
       5 #define MAX_BLOCK_SIZE 100 // Maximum size of each block
                                                                                 Enter block index to access (0 to 3) or -1 to exit: 2
9
       7 // Structure to represent a file
                                                                                 Accessing block 2: Block 2: Data for the third block.
       8 → typedef struct {
             char *indexBlock[MAX_BLOCKS]; // Array of pointers to data
                                                                                 Enter block index to access (0 to 3) or -1 to exit: 0
                                                                                 Accessing block 0: Block 0: Data for the first block.
                 blocks
0
              int numBlocks;
      10
                                            // Number of blocks currently used
                  in the file
                                                                                 Enter block index to access (0 to 3) or -1 to exit: -1
      11 } File;
                                                                                 Memory released.
(
      12 // Function to initialize the file with no data
       13 - void initializeFile(File *file) {
0
             file->numBlocks = 0;
                                                                                 === Code Execution Successful ===
              for (int i = 0; i < MAX_BLOCKS; i++) {
      15 -
JS
                  file->indexBlock[i] = NULL;
      16
      17
      18 }
      19 // Function to add data to a file (allocate a block)
```









```
®<sub>⊚</sub> Execute
              Current Permissions of testfile.txt:
                                                                   File Permissions: rw- r-- r--
                                                                   Permissions changed successfully.
                                                                   Updated Permissions of testfile.txt:
                                                                   File Permissions: rwx r-x r-x
  6 void display_permissions(mode_t mode) {
                                                                   Owner and Group changed successfully.
         printf("File Permissions: ");
         printf("%c", (mode & S_IRUSR) ? 'r' : '-');
         printf("%c", (mode & S_IWUSR) ? 'w' : '-');
 10
         printf("%c ", (mode & S_IXUSR) ? 'x' : '-');
         printf("%c", (mode & 5_IRGRP) ? 'r' : '-');
         printf("%c", (mode & S_IWGRP) ? 'w' : '-');
         printf("%c ", (mode & S_IXGRP) ? 'x' : '-');
         printf("%c", (mode & S_IROTH) ? 'r' : '-');
         printf("%c", (mode & S_IWOTH) ? 'w' : '-');
         printf("%c\n", (mode & S_IXOTH) ? 'x' : '-');
     }
 21 void change_permissions(const char* filename, mode_t mode)
         if (chmod(filename, mode) == -1) {
```

```
® Execute ⟨/> Source Code | ≪ Share (?) Help
                                                                     File 'sample.txt' opened successfully.
                                                                     File access mode: Read/Write
                                                                     File pointer moved to the end: 0 bytes
                                                                     File Information:
                                                                     Size: 0 bytes
                                                                     Permissions: rw-
  8 - int main() {
          int fd;
                                                                     Directory Contents of '.':
          struct stat fileStat;
 10
         DIR *dir;
         struct dirent *entry;
                                                                     -> sample.txt
                                                                     -> main
         const char *filename = "sample.txt";
                                                                     -> testfile.txt
         const char *dirname = "."; // Current directory
                                                                     -> main.c
          fd = open(filename, O_RDWR | O_CREAT, 0644);
          if (fd == -1) {
             perror("Error opening file");
 20
             return 1;
          printf("File '%s' opened successfully.\n", filename);
          int flags = fcntl(fd, F GETFL):
```

```
®<sub>⊚</sub> Execute
                                                                   Enter the number of processes: 4
                                                                   Enter Arrival Time, Burst Time and Priority for each process:
     typedef struct {
                                                                   Process 1: 0 8 2
         int pid;
                                                                   Process 2: 1 4 1
         int arrival_time;
                                                                   Process 3: 2 8 3
         int burst_time;
                                                                   Process 4: 3 5 2
         int priority;
                                                                   PID AT BT PR CT TAT WT
         int remaining_time;
                                                                                   12 12 4
         int completion_time;
                                                                   2
         int turnaround_time;
  10
                                                                       2
                                                                           8
                                                                                   25 23 15
         int waiting_time;
                                                                                   17 14 9
                                                                               2
     } Process;
     int main() {
                                                                   Average Turnaround Time: 13.25
         Process p[MAX];
                                                                   Average Waiting Time: 7.00
         int n, time = \theta, completed = \theta;
         int i, current = -1, prev = -1;
         float total_tat = 0, total_wt = 0;
         printf("Enter the number of processes: ");
         scanf("%d", &n);
         printf("Enter Arrival Time, Burst Time and Priority
             for each process:\n");
         for (i = 0; i < n; i++) {
             p[i].pid = i + 1;
```

