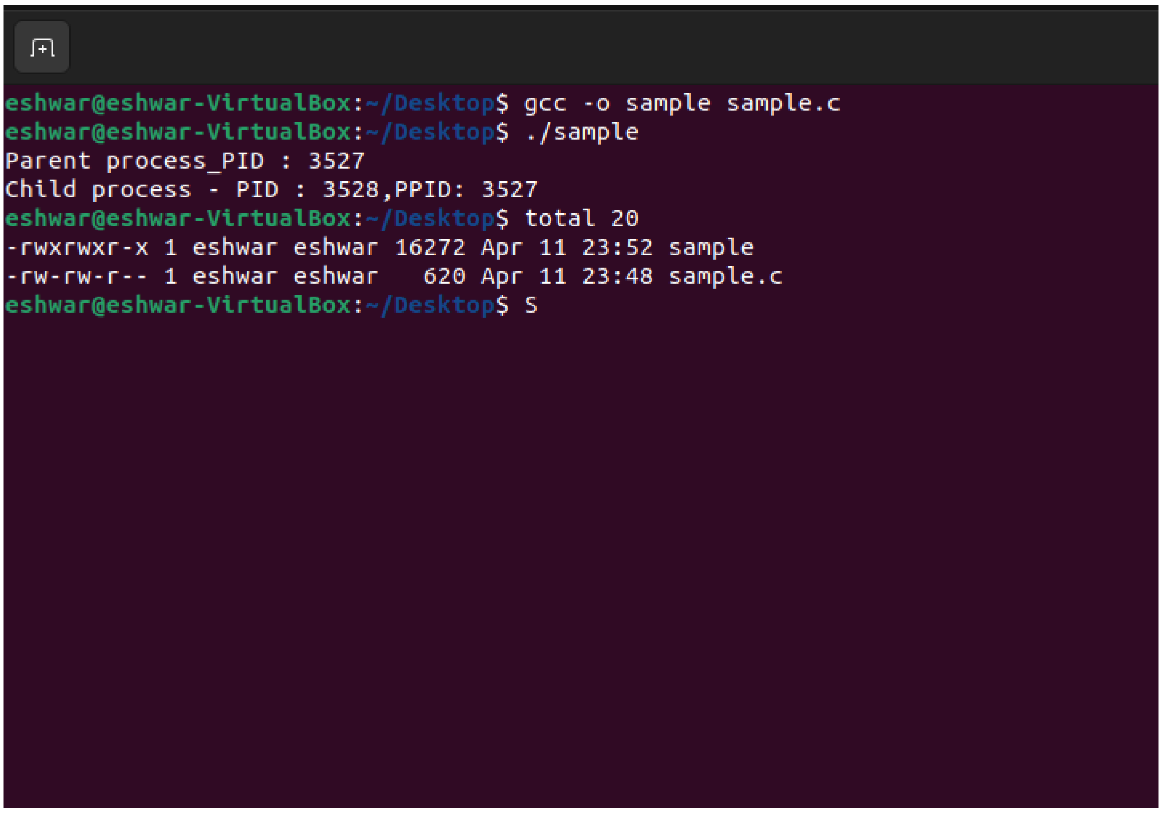
Program:

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

int main() {

   pid_t parent_pid = getpid();
   printf("Parent process_PID : %d\n",parent_pid);
   pid_t child_pid = fork();

   printf("System Calls");

   if(child_pid == -1)
   {
     perror("Fork failed");
     exit(EXIT_FAILURE);
     }

   if(child_pid == 0)
   {
     printf("Child process - PID : %d,PPID: %d \n",getpid(),getppid());
     char *cmd[]={"ls","-l",NULL};
     if(execvp(cmd[0],cmd)== -1)
     {
        perror("execvp failed");
        exit(EXIT_FAILURE);
        }
      }

   return 0;
}
```

Output:

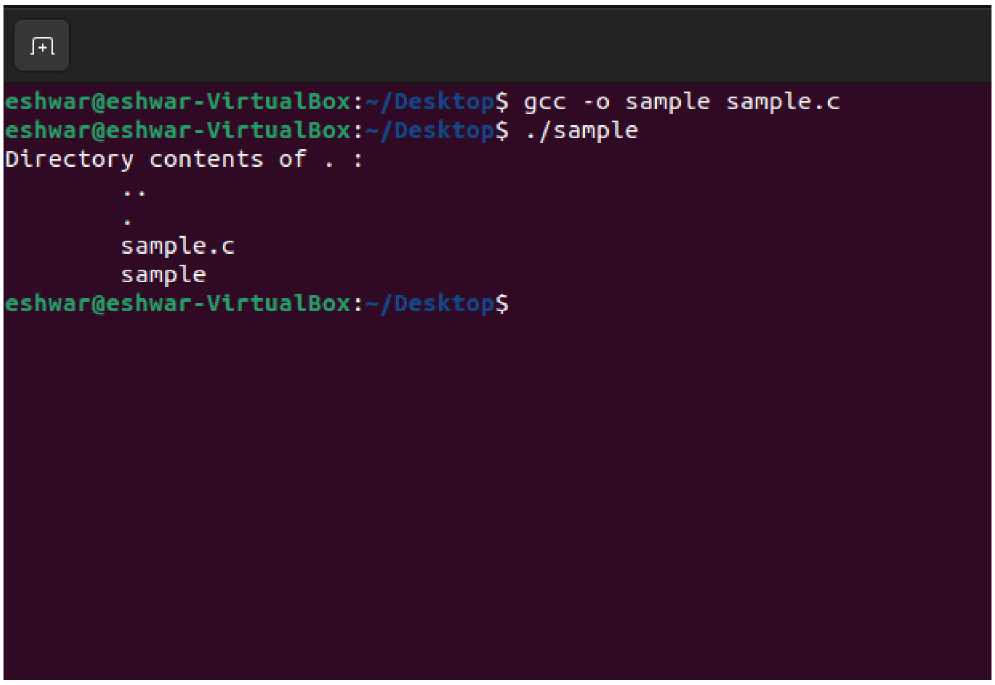


```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Parent process_PID : 3527
Child process - PID : 3528,PPID: 3527
eshwar@eshwar-VirtualBox:~/Desktop$ total 20
-rwxrwxr-x 1 eshwar eshwar 16272 Apr 11 23:52 sample
-rw-rw-r-- 1 eshwar eshwar   620 Apr 11 23:48 sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ S
```

Program:

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

int main(int argc, char *argv[]) {
   const char *dir_path = (argc > 1) ? argv[1]: ".";
   DIR *dp = opendir(dir_path);

printf("System Calls");

   if (dp == NULL) {
      perror("opendir failed");
      exit(EXIT_FAILURE);
   }

   struct dirent *entry;
   printf("Directory contents of %s : \n",dir_path);

   while ((entry = readdir(dp)) != NULL) {
      printf("\t%s \n", entry->d_name);
   }

   closedir(dp);

   return EXIT_SUCCESS;
}
```

Output:

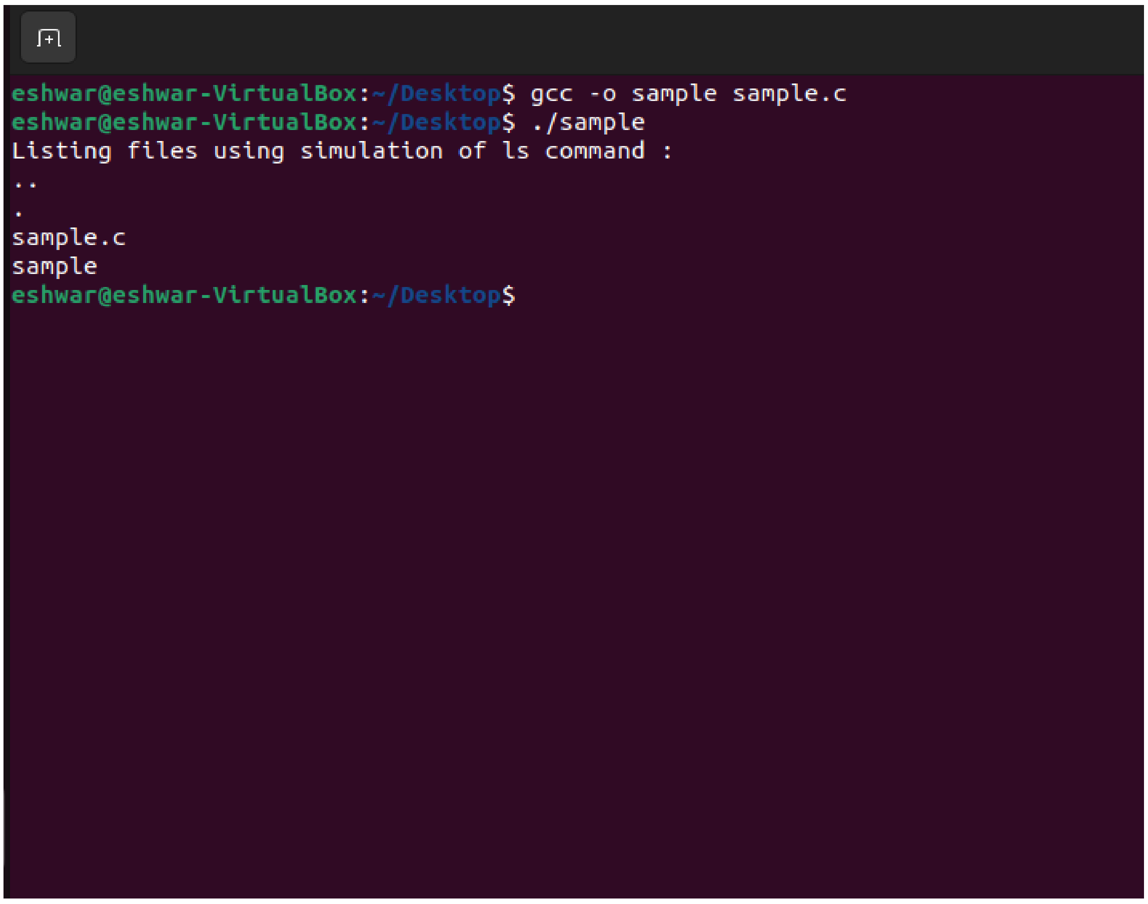


```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Directory contents of . :
        ..
        .
        sample.c
        sample
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

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

int main(int argc, char **argv) {
   DIR *dp;
   struct dirent *link;

   if (argc != 2) {
      printf("Usage: %s <directory>\n", argv[0]);
      return 1;
   }

   dp = opendir(argv[1]);
   if (dp == NULL) {
      perror("opendir");
      return 1;
   }

   printf("\nContents of the directory %s are:\n", argv[1]);
   while ((link = readdir(dp)) != NULL) {
      printf("%s\n", link->d_name);
   }

   closedir(dp);

   return 0;
}
```

Output:

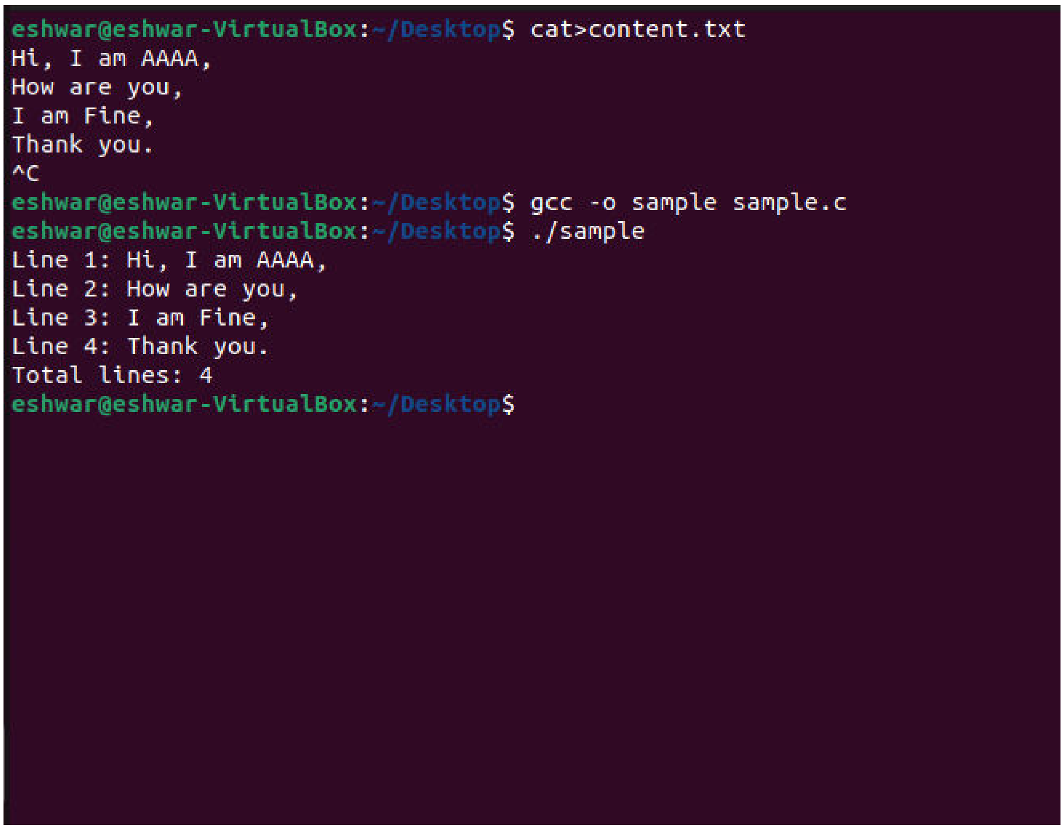


```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Listing files using simulation of ls command :
..
.
sample.c
sample
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h> // for exit()

#define MAX 1024
void usage() {
   printf("Usage:\t./a.out filename word\n");
}

int main(int argc, char *argv[]) {
   FILE *fp;
   char fline[MAX];
   char *newline;
   int count = 0;
   int occurrences = 0;

   if (argc != 3) {
      usage();
      exit(1);
   }

   if (!(fp = fopen(argv[1], "r"))) {
      printf("grep: could not open file: %s\n", argv[1]);
      exit(1);
   }

   while (fgets(fline, MAX, fp) != NULL) {
      count++;
      if ((newline = strchr(fline, '\n'))) {
         *newline = '\0';
      }
      if (strstr(fline, argv[2]) != NULL) {
         printf("%s: %d %s\n", argv[1], count, fline);
         occurrences++;
      }
   }
   fclose(fp);
   if (occurrences == 0) {
      printf("No occurrences found for '%s' in file '%s'\n", argv[2], argv[1]);
   }
   return 0;
}
```

Output:

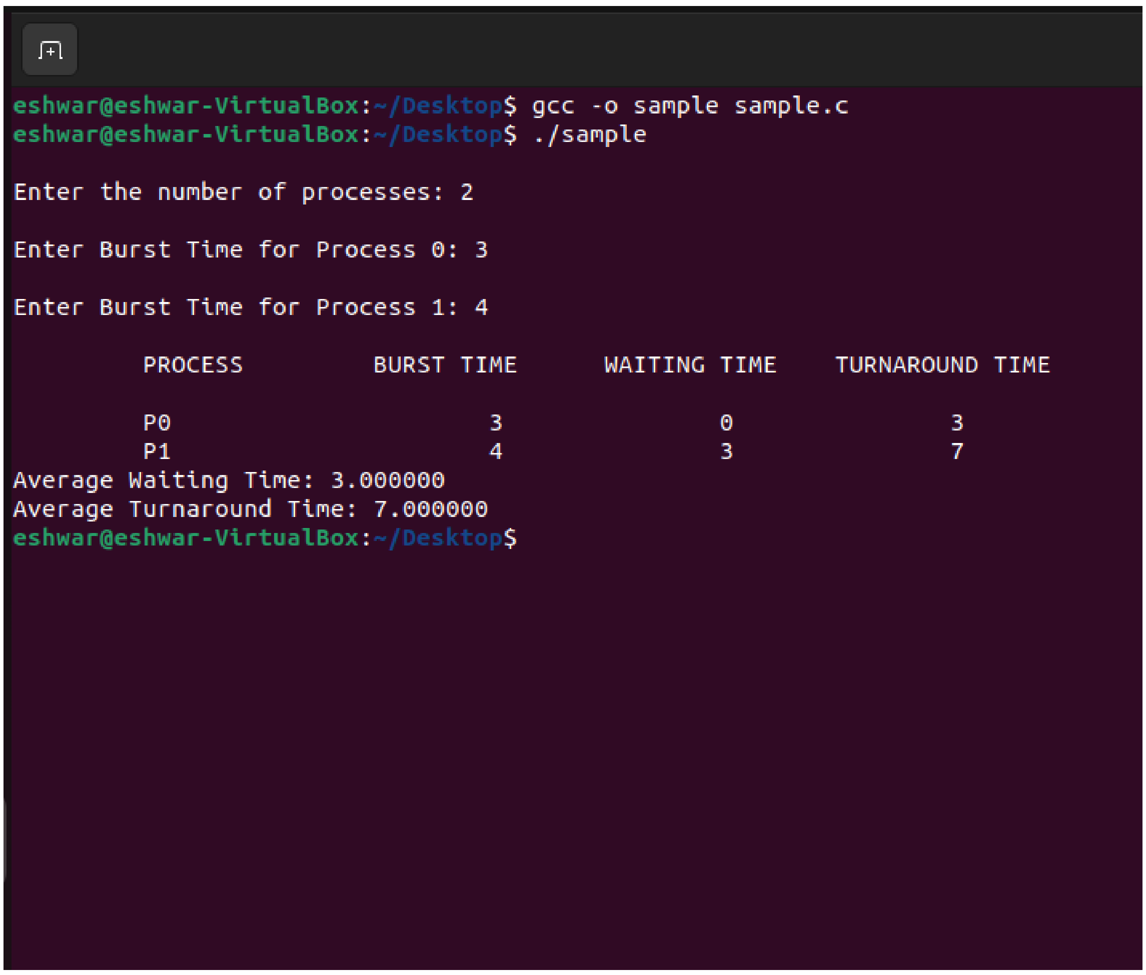


```
eshwar@eshwar-VirtualBox:~/Desktop$ cat>content.txt
Hi, I am AAAA,
How are you,
I am Fine,
Thank you.
^C
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Line 1: Hi, I am AAAA,
Line 2: How are you,
Line 3: I am Fine,
Line 4: Thank you.
Total lines: 4
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

```
#include <stdio.h>

int main() {
   int bt[20], wt[20], tat[20], i, n;
   float wtavg = 0, tatavg = 0;
printf(“CPU Scheduling Algorithm – FIFS ”);

   printf("\nEnter the number of processes: ");
   scanf("%d", &n);

   for (i = 0; i < n; i++) {
      printf("\nEnter Burst Time for Process %d: ", i);
      scanf("%d", &bt[i]);
   }

   wt[0] = 0;
   tat[0] = bt[0];

   for (i = 1; i < n; i++) {
      wt[i] = wt[i - 1] + bt[i - 1];
      tat[i] = tat[i - 1] + bt[i];
      wtavg += wt[i];
      tatavg += tat[i];
   }

   printf("\n\t PROCESS \t BURST TIME \t WAITING TIME \t TURNAROUND TIME\n");
   for (i = 0; i < n; i++) {
      printf("\n\t P%d \t\t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
   }

   printf("\nAverage Waiting Time: %f", wtavg);
   printf("\nAverage Turnaround Time: %f\n", tatavg );

   return 0;
}
```

Output:

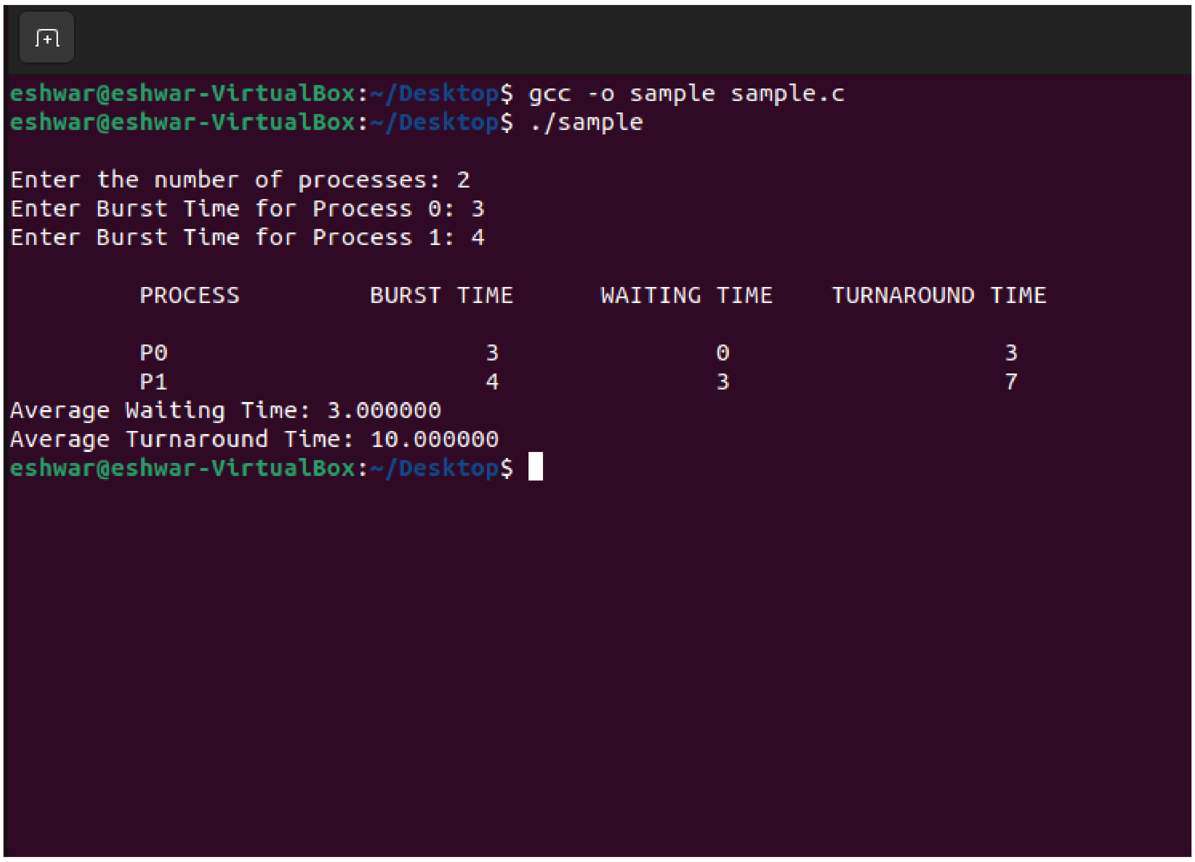


```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample

Enter the number of processes: 2

Enter Burst Time for Process 0: 3

Enter Burst Time for Process 1: 4

         PROCESS          BURST TIME      WAITING TIME    TURNAROUND TIME

         P0                       3                0                3
         P1                       4                3                7
Average Waiting Time: 3.000000
Average Turnaround Time: 7.000000
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

```
#include <stdio.h>

int main() {
   int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
   float wtavg = 0, tatavg = 0;
printf(“CPU Scheduling Algorithm - SJF”);

   printf("\nEnter the number of processes: ");
   scanf("%d", &n);

   for (i = 0; i < n; i++) {
      p[i] = i;
      printf("Enter Burst Time for Process %d: ", i);
      scanf("%d", &bt[i]);
   }

   for (i = 0; i < n; i++) {
      for (k = i + 1; k < n; k++) {
         if (bt[i] > bt[k]) {
            temp = bt[i];
            bt[i] = bt[k];
            bt[k] = temp;

            temp = p[i];
            p[i] = p[k];
            p[k] = temp;
         }
      }
   }

   wt[0] = wtavg = 0;
   tat[0] = tatavg = bt[0];

   for (i = 1; i < n; i++) {
      wt[i] = wt[i - 1] + bt[i - 1];
      tat[i] = tat[i - 1] + bt[i];
      wtavg += wt[i];
      tatavg += tat[i];
   }

   printf("\n\t PROCESS \t BURST TIME \t WAITING TIME \t TURNAROUND TIME\n");
   for (i = 0; i < n; i++) {
        printf("\n\t P%d \t\t\t %d \t\t\t %d \t\t\t %d", p[i], bt[i], wt[i], tat[i]);
    }

    printf("\nAverage Waiting Time: %f", wtavg );
    printf("\nAverage Turnaround Time: %f\n", tatavg );

    return 0;
}
```

Output:

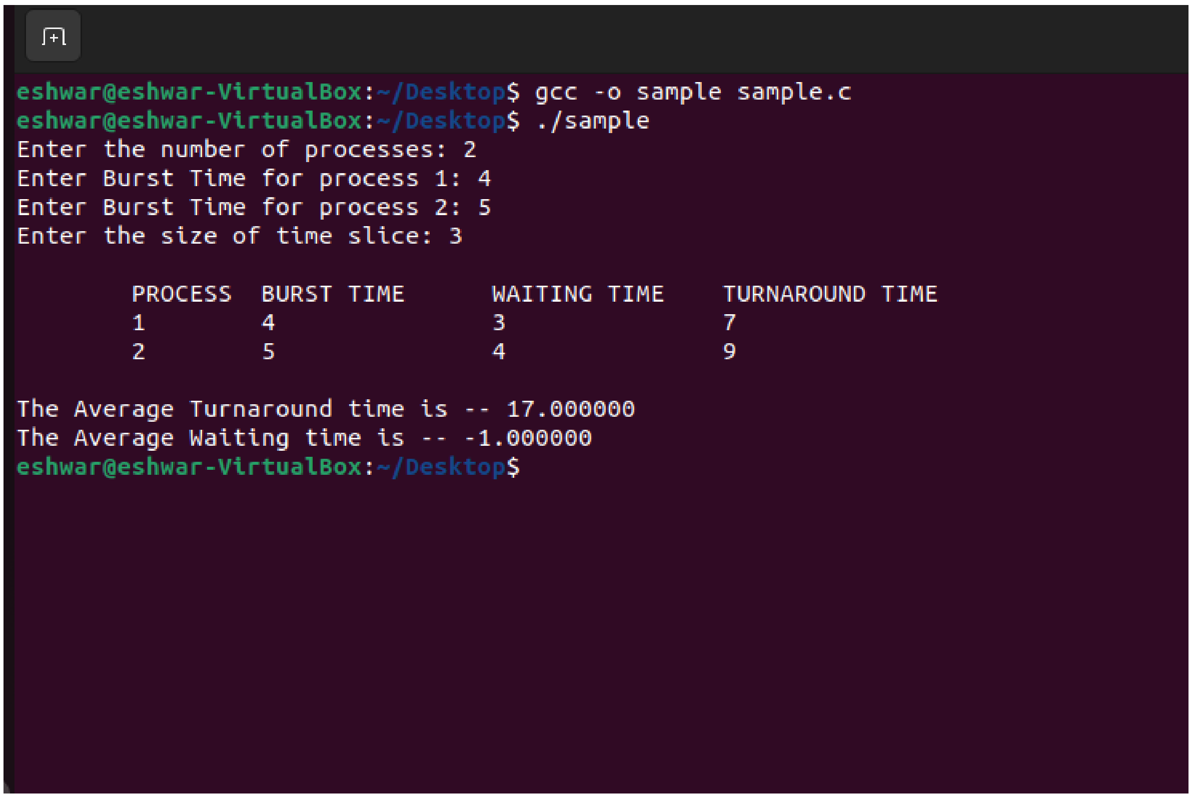


```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample

Enter the number of processes: 2
Enter Burst Time for Process 0: 3
Enter Burst Time for Process 1: 4

         PROCESS          BURST TIME      WAITING TIME    TURNAROUND TIME

         P0                       3               0                   3
         P1                       4               3                   7
Average Waiting Time: 3.000000
Average Turnaround Time: 10.000000
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

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

int main() {
    int i, j, n, bu[10], wa[10], tat[10], ct[10], max, t;
    float awt = 0, att = 0, temp = 0;
    printf("CPU Scheduling Algorithm – Round Robin");
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        printf("Enter Burst Time for process %d: ", i + 1);
        scanf("%d", &bu[i]);
        ct[i] = bu[i];
    }

    printf("Enter the size of time slice: ");
    scanf("%d", &t);

    max = bu[0];

    for (i = 1; i < n; i++)
        if (max < bu[i])
            max = bu[i];

    for (j = 0; j < (max / t) + 1; j++) {
        for (i = 0; i < n; i++) {
            if (bu[i] != 0) {
                if (bu[i] <= t) {
                    tat[i] = temp + bu[i];
                    temp += bu[i];
                    bu[i] = 0;
                } else {
                    bu[i] -= t;
                    temp += t;
                }

                wa[i] = tat[i] - ct[i];
                att += tat[i];
                awt += wa[i];
            }
        }
    }

printf("\n\tPROCESS\t BURST TIME \t WAITING TIME \t TURNAROUND TIME\n");

    for (i = 0; i < n; i++)
        printf("\t%d \t %d \t\t %d \t\t %d \n", i + 1, ct[i], wa[i], tat[i]);

printf("\nThe Average Turnaround time is -- %f", att );
printf("\nThe Average Waiting time is -- %f\n", awt );


    return 0;
}
```

Output:



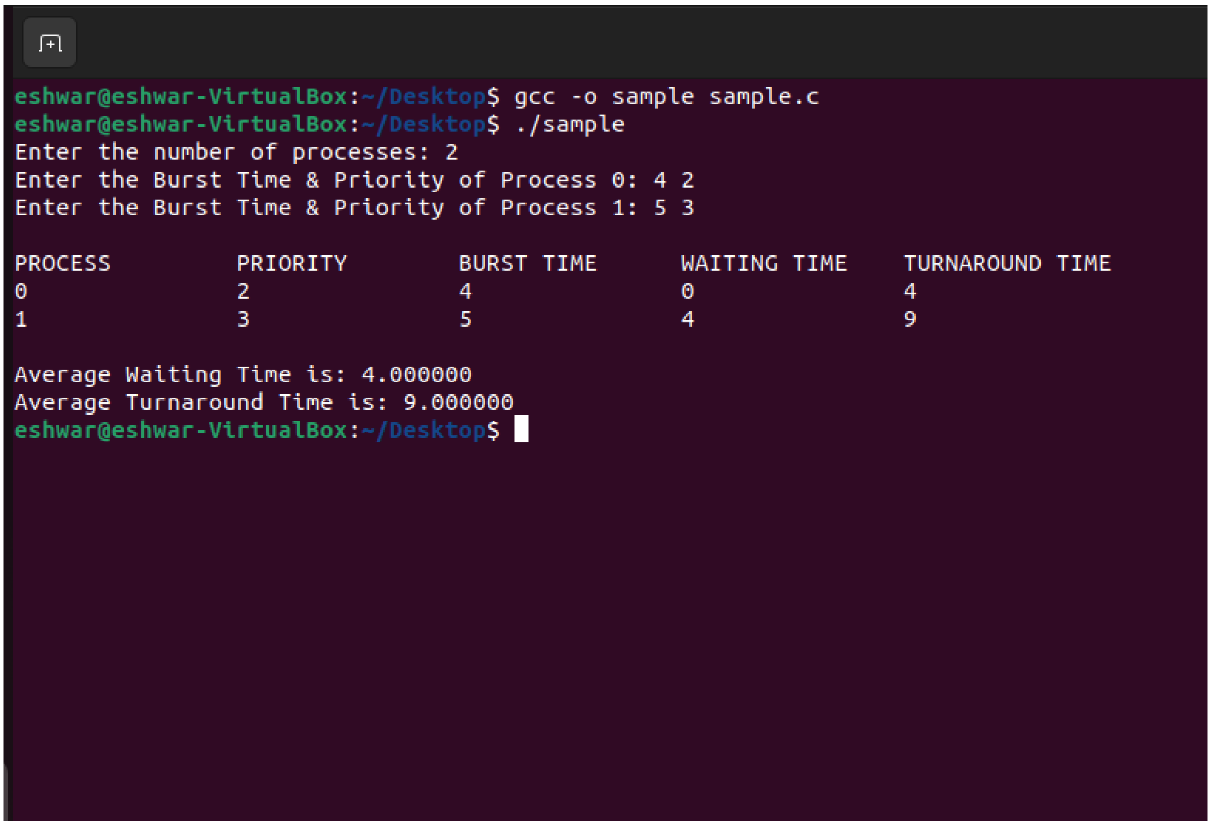
```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Enter the number of processes: 2
Enter Burst Time for process 1: 4
Enter Burst Time for process 2: 5
Enter the size of time slice: 3

        PROCESS  BURST TIME      WAITING TIME    TURNAROUND TIME
        1        4               3               7
        2        5               4               9

The Average Turnaround time is -- 17.000000
The Average Waiting time is -- -1.000000
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

```
#include<stdio.h>

int main() {
   int p[20], bt[20], pri[20], wt[20], tat[20], i, k, n, temp;
   float wtavg = 0, tatavg = 0;
   printf(“CPU Scheduling Algorithm - Priority”);
   printf("Enter the number of processes: ");
   scanf("%d", &n);

   for(i = 0; i < n; i++) {
      p[i] = i;
      printf("Enter the Burst Time & Priority of Process %d: ", i);
      scanf("%d %d", &bt[i], &pri[i]);
   }

   for(i = 0; i < n; i++) {
      for(k = i + 1; k < n; k++) {
         if(pri[i] > pri[k]) {
            temp = p[i];
            p[i] = p[k];
            p[k] = temp;

            temp = bt[i];
            bt[i] = bt[k];
            bt[k] = temp;

            temp = pri[i];
            pri[i] = pri[k];
            pri[k] = temp;
         }
      }
   }

   wt[0] = 0;
   tat[0] = bt[0];

   for(i = 1; i < n; i++) {
      wt[i] = wt[i - 1] + bt[i - 1];
      tat[i] = tat[i - 1] + bt[i];
      wtavg += wt[i];
      tatavg += tat[i];
   }
```

```
    printf("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROUND TIME\n");

    for(i = 0; i < n; i++)
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i], pri[i], bt[i], wt[i], tat[i]);

    printf("\nAverage Waiting Time is: %f", wtavg );
    printf("\nAverage Turnaround Time is: %f\n", tatavg );

    return 0;
}
```

Output:

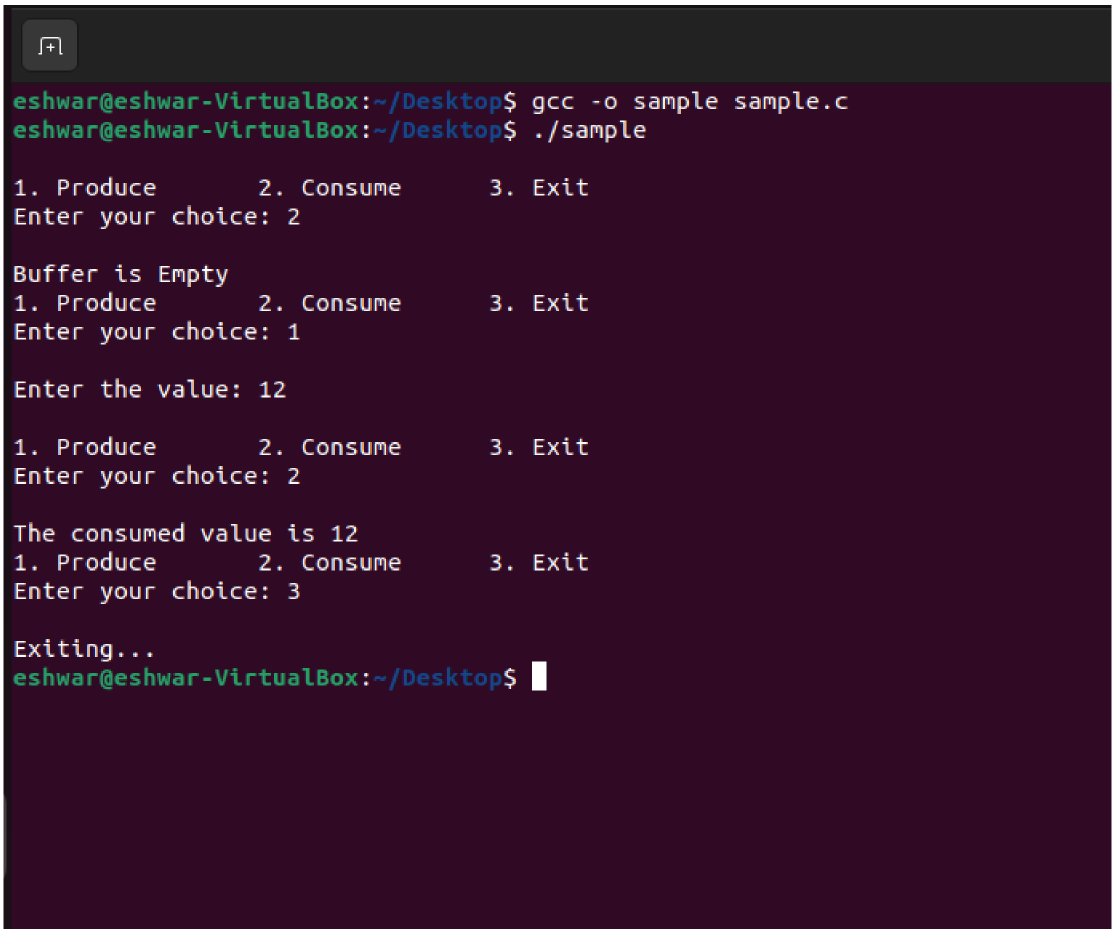


```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Enter the number of processes: 2
Enter the Burst Time & Priority of Process 0: 4 2
Enter the Burst Time & Priority of Process 1: 5 3

PROCESS         PRIORITY        BURST TIME      WAITING TIME    TURNAROUND TIME
0               2               4               0               4
1               3               5               4               9

Average Waiting Time is: 4.000000
Average Turnaround Time is: 9.000000
eshwar@eshwar-VirtualBox:~/Desktop$ 
```

Program:

```
#include <stdio.h>

int main() {
   int buffer[10], bufsize = 10, in = 0, out = 0, produce, consume, choice = 0;
printf(“Producer Consumer Problem”);
   while (choice != 3) {
      printf("\n1. Produce \t 2. Consume \t 3. Exit");
      printf("\nEnter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            if ((in + 1) % bufsize == out)
               printf("\nBuffer is Full");
            else {
               printf("\nEnter the value: ");
               scanf("%d", &produce);
               buffer[in] = produce;
               in = (in + 1) % bufsize;
            }
            break;

         case 2:
            if (in == out)
               printf("\nBuffer is Empty");
            else {
               consume = buffer[out];
               printf("\nThe consumed value is %d", consume);
               out = (out + 1) % bufsize;
            }
            break;

         case 3:
            printf("\nExiting...\n");
            break;

         default:
            printf("\nInvalid choice!\n");
            break;
      }
   }
   return 0;
}
```

Output:

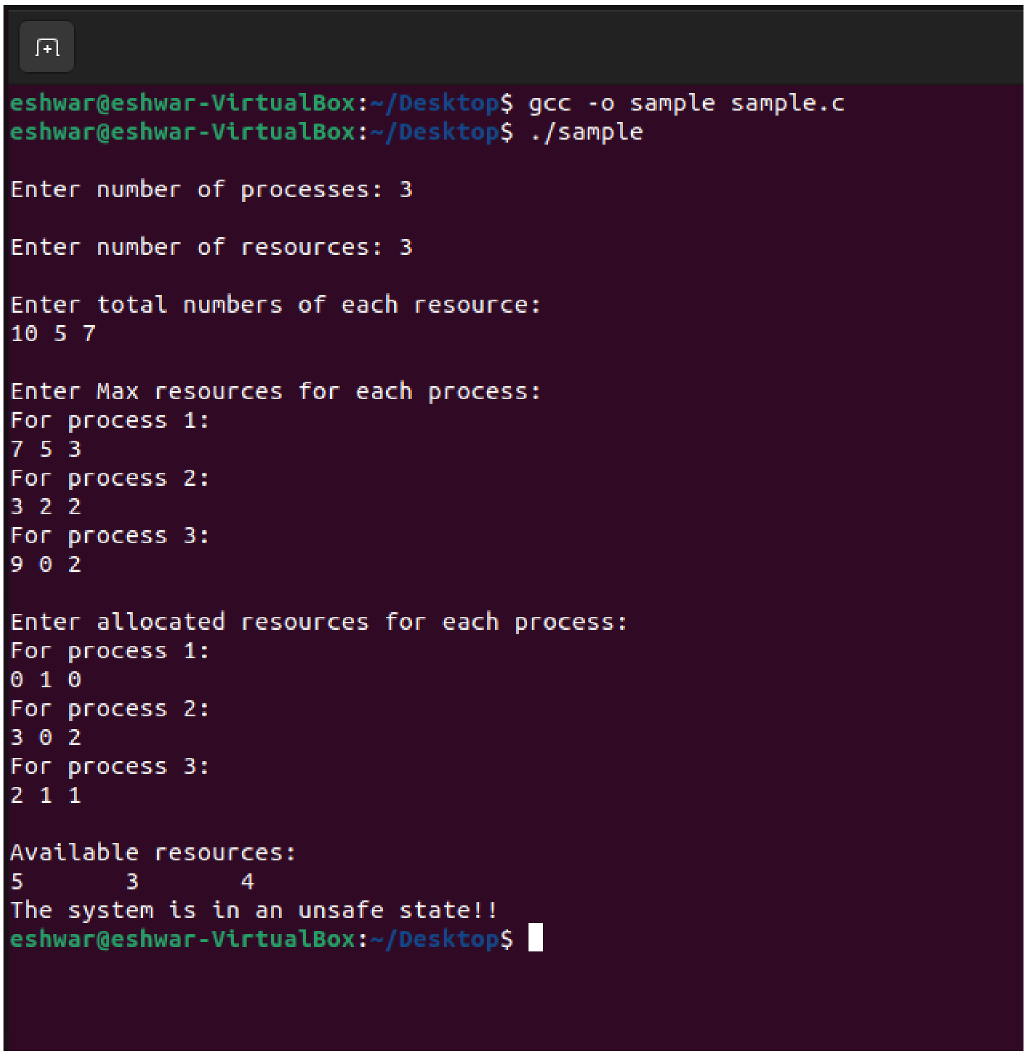


```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample

1. Produce       2. Consume      3. Exit
Enter your choice: 2

Buffer is Empty
1. Produce       2. Consume      3. Exit
Enter your choice: 1

Enter the value: 12

1. Produce       2. Consume      3. Exit
Enter your choice: 2

The consumed value is 12
1. Produce       2. Consume      3. Exit
Enter your choice: 3

Exiting...
eshwar@eshwar-VirtualBox:~/Desktop$ 
```

Program:

```
#include <stdio.h>

int main() {
    int pno, rno, i, j, prc, count = 0, t, total;
printf(“Bankers Problem”);
    printf("\nEnter number of processes: ");
    scanf("%d", &pno);

    printf("\nEnter number of resources: ");
    scanf("%d", &rno);

    int max[pno][rno], allocated[pno][rno], need[pno][rno], avail[rno], work[rno], flag[pno];

    printf("\nEnter total numbers of each resource:\n");
    for (i = 0; i < rno; i++)
        scanf("%d", &avail[i]);

    printf("\nEnter Max resources for each process:\n");
    for (i = 0; i < pno; i++) {
        printf("For process %d:\n", i + 1);
        for (j = 0; j < rno; j++)
            scanf("%d", &max[i][j]);
    }

    printf("\nEnter allocated resources for each process:\n");
    for (i = 0; i < pno; i++) {
        printf("For process %d:\n", i + 1);
        for (j = 0; j < rno; j++)
            scanf("%d", &allocated[i][j]);
    }

    // Calculating the need matrix
    for (i = 0; i < pno; i++) {
        for (j = 0; j < rno; j++) {
            need[i][j] = max[i][j] - allocated[i][j];
        }
    }

    printf("\nAvailable resources:\n");
    for (j = 0; j < rno; j++) {
        total = 0;
        for (i = 0; i < pno; i++) {
            total += allocated[i][j];
        }
        avail[j] -= total;
        work[j] = avail[j];
        printf("%d\t", work[j]);
    }

    do {
        prc = -1;
        for (i = 0; i < pno; i++) {
            if (flag[i] == 0) {
                prc = i;
                for (j = 0; j < rno; j++) {
                    if (work[j] < need[i][j]) {
                        prc = -1;
                        break;
                    }
                }
                if (prc != -1) break;
            }
        }
        if (prc != -1) {
            printf("\nProcess %d completed\n", prc + 1);
            count++;
            for (j = 0; j < rno; j++) {
                work[j] += allocated[prc][j];
                allocated[prc][j] = 0;
                max[prc][j] = 0;
                flag[prc] = 1;
            }
        }
    } while (count != pno && prc != -1);

    if (count == pno)
        printf("\nThe system is in a safe state!!");
    else
        printf("\nThe system is in an unsafe state!!");

    return 0;
}
```

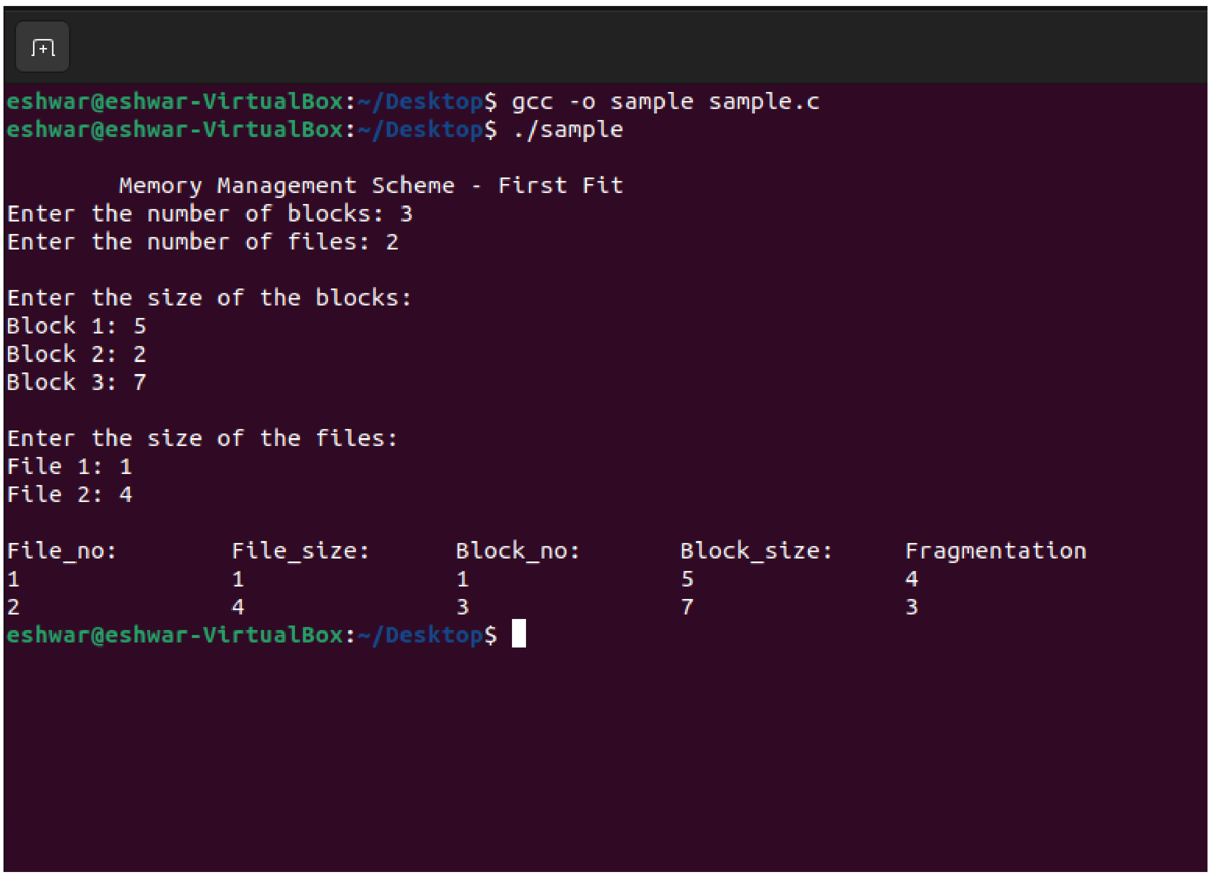
Output:



```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample

Enter number of processes: 3

Enter number of resources: 3

Enter total numbers of each resource:
10 5 7

Enter Max resources for each process:
For process 1:
7 5 3
For process 2:
3 2 2
For process 3:
9 0 2

Enter allocated resources for each process:
For process 1:
0 1 0
For process 2:
3 0 2
For process 3:
2 1 1

Available resources:
5       3       4
The system is in an unsafe state!!
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:
```
#include<stdio.h>
#define max 25

void main() {
   int frag[max], b[max], f[max], i, j, nb, nf, temp;
   static int bf[max], ff[max];

   printf("\n\tMemory Management Scheme - First Fit\n");
   printf("Enter the number of blocks: ");
   scanf("%d", &nb);
   printf("Enter the number of files: ");
   scanf("%d", &nf);

   printf("\nEnter the size of the blocks:\n");
   for (i = 1; i <= nb; i++) {
      printf("Block %d: ", i);
      scanf("%d", &b[i]);
   }

   printf("\nEnter the size of the files:\n");
   for (i = 1; i <= nf; i++) {
      printf("File %d: ", i);
      scanf("%d", &f[i]);
   }

   printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragmentation\n");
   for (i = 1; i <= nf; i++) {
      for (j = 1; j <= nb; j++) {
         if (bf[j] != 1) {
            temp = b[j] - f[i];
            if (temp >= 0)
               break;
         }
      }
      frag[i] = temp;
      bf[j] = 1;
      ff[i] = j;


      printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i, f[i], ff[i], b[ff[i]], frag[i]);
   }

}
```

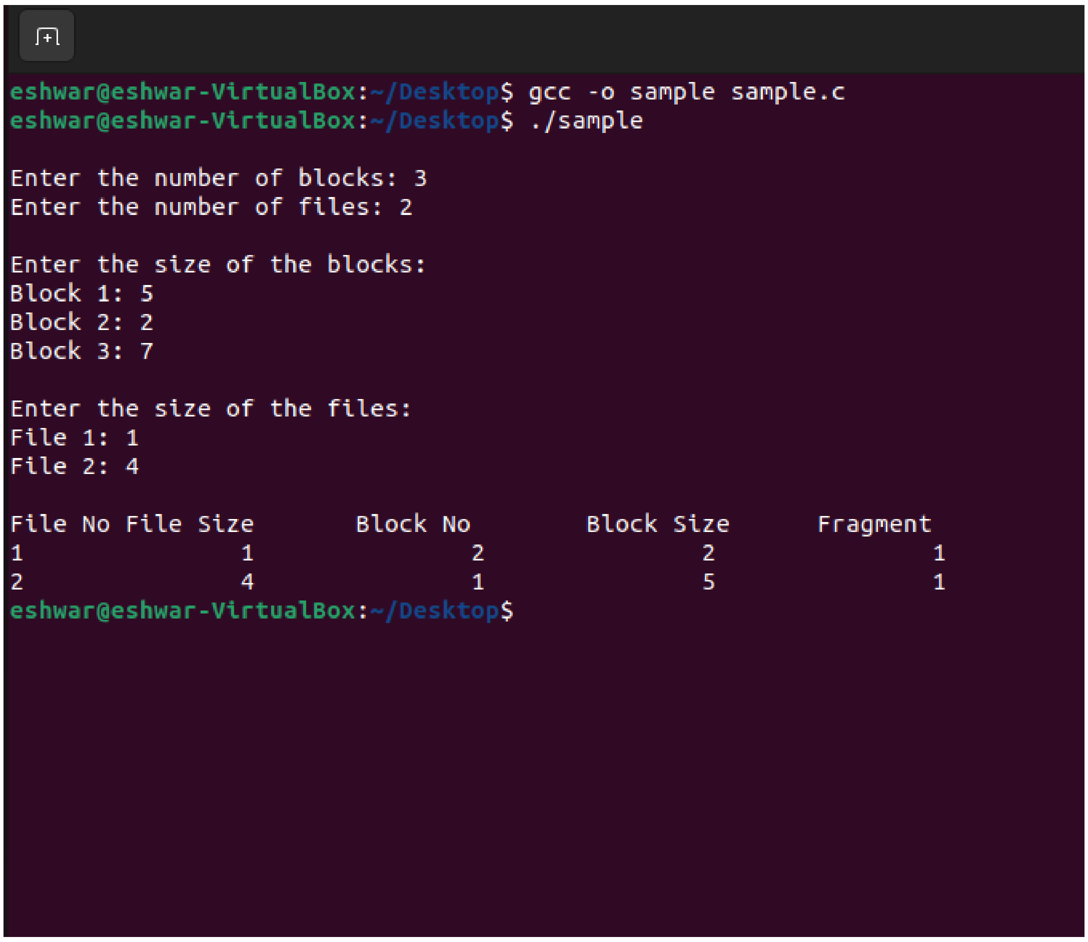
Output:



```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample

        Memory Management Scheme - First Fit
Enter the number of blocks: 3
Enter the number of files: 2

Enter the size of the blocks:
Block 1: 5
Block 2: 2
Block 3: 7

Enter the size of the files:
File 1: 1
File 2: 4

File_no:        File_size:      Block_no:       Block_size:     Fragmentation
1               1               1               5               4
2               4               3               7               3
eshwar@eshwar-VirtualBox:~/Desktop$ 
```

Program:
```
#include<stdio.h>
#define max 25

void main() {
   int frag[max], b[max], f[max], i, j, nb, nf, temp, lowest;
   static int bf[max], ff[max];
printf("\n\tMemory Management Scheme - Best Fit\n");

   printf("\nEnter the number of blocks: ");
   scanf("%d", &nb);

   printf("Enter the number of files: ");
   scanf("%d", &nf);

   printf("\nEnter the size of the blocks:\n");
   for (i = 1; i <= nb; i++) {
      printf("Block %d: ", i);
      scanf("%d", &b[i]);
   }

   printf("\nEnter the size of the files:\n");
   for (i = 1; i <= nf; i++) {
      printf("File %d: ", i);
      scanf("%d", &f[i]);
   }
   printf("\nFile No\tFile Size\tBlock No\tBlock Size\tFragment\n");
   for (i = 1; i <= nf; i++) {
      lowest = 10000;
      for (j = 1; j <= nb; j++) {
         if (bf[j] != 1) {
            temp = b[j] - f[i];
            if (temp >= 0 && lowest > temp) {
               ff[i] = j;
               lowest = temp;
            }
         }
      }
      frag[i] = lowest;
      bf[ff[i]] = 1;

      printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i, f[i], ff[i], b[ff[i]], frag[i]);
   }
}
```

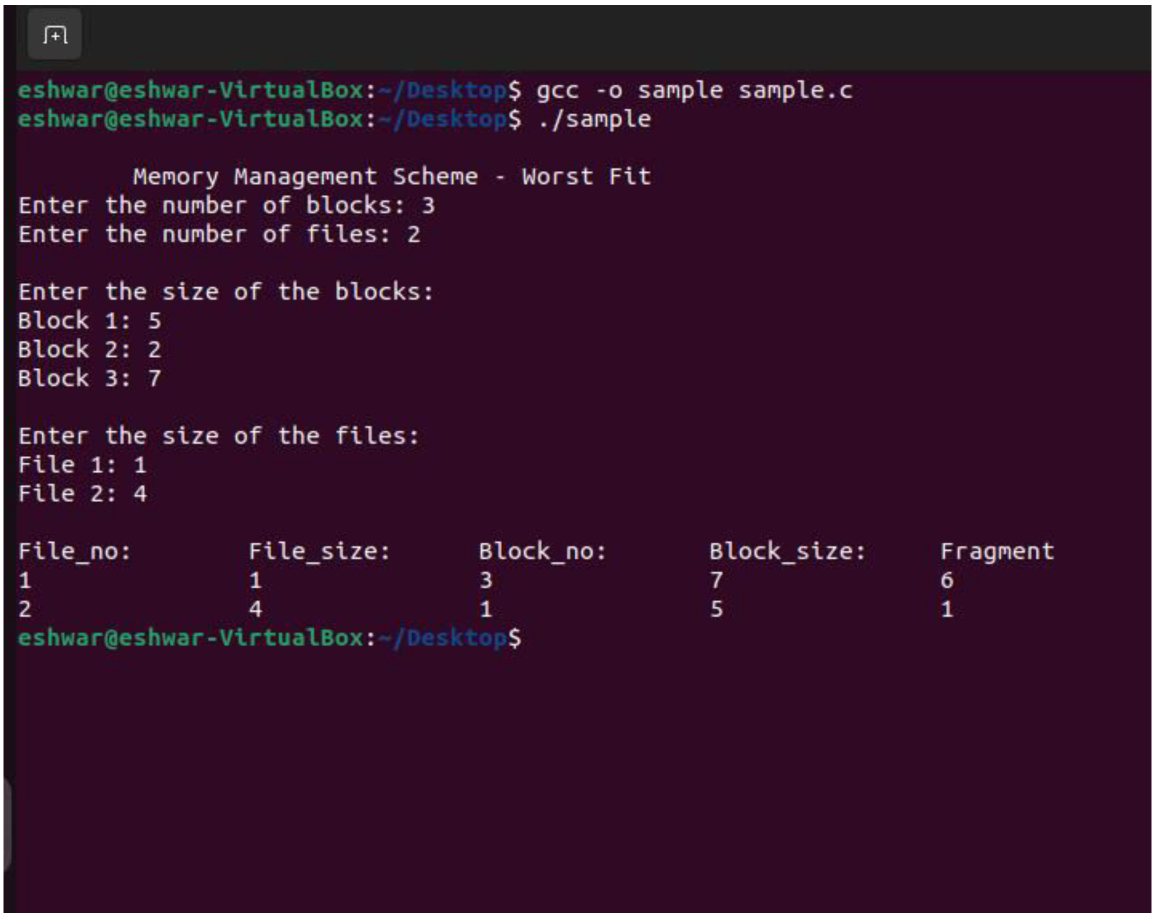
Output:



```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample

Enter the number of blocks: 3
Enter the number of files: 2

Enter the size of the blocks:
Block 1: 5
Block 2: 2
Block 3: 7

Enter the size of the files:
File 1: 1
File 2: 4

File No File Size       Block No        Block Size      Fragment
1               1               2               2               1
2               4               1               5               1
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:
```
#include<stdio.h>
#define max 25

void main() {
   int frag[max], b[max], f[max], i, j, nb, nf, temp, highest = 0;
   static int bf[max], ff[max];

   printf("\n\tMemory Management Scheme - Worst Fit\n");
   printf("Enter the number of blocks: ");
   scanf("%d", &nb);

   printf("Enter the number of files: ");
   scanf("%d", &nf);

   printf("\nEnter the size of the blocks:\n");
   for (i = 1; i <= nb; i++) {
      printf("Block %d: ", i);
      scanf("%d", &b[i]);
   }

   printf("\nEnter the size of the files:\n");
   for (i = 1; i <= nf; i++) {
      printf("File %d: ", i);
      scanf("%d", &f[i]);
   }

   printf("\nFile_no:\tFile_size:\tBlock_no:\tBlock_size:\tFragment\n");
   for (i = 1; i <= nf; i++) {
      highest = 0;
      for (j = 1; j <= nb; j++) {
         if (bf[j] != 1) {
            temp = b[j] - f[i];
            if (temp >= 0 && highest < temp) {
               highest = temp;
               ff[i] = j;
            }
         }
      }
      frag[i] = highest;
      bf[ff[i]] = 1;
      printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i, f[i], ff[i], b[ff[i]], frag[i]);
   }
}
```

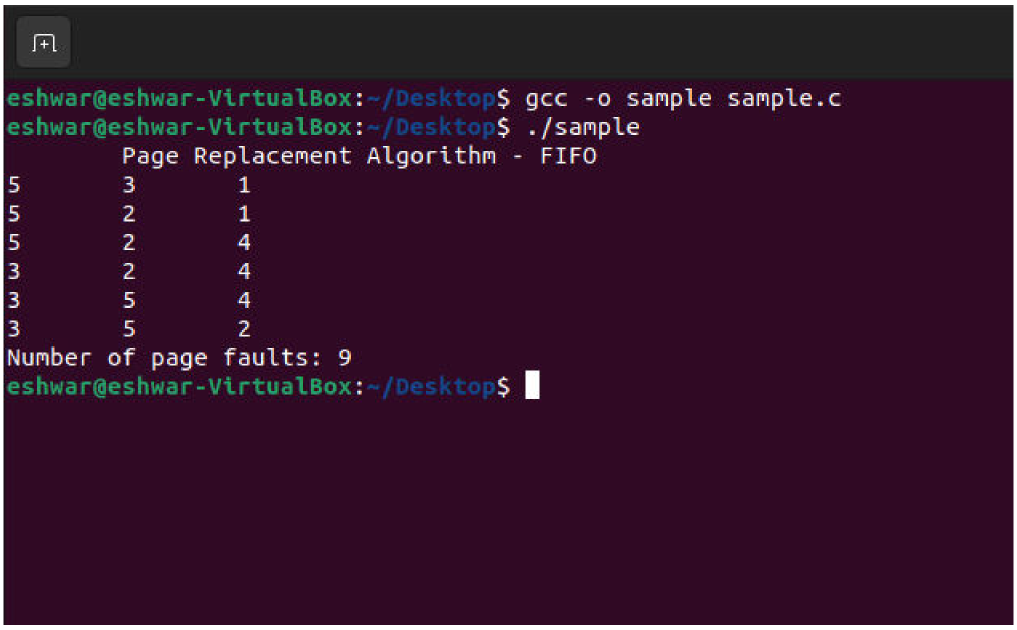
Output:



```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample

        Memory Management Scheme - Worst Fit
Enter the number of blocks: 3
Enter the number of files: 2

Enter the size of the blocks:
Block 1: 5
Block 2: 2
Block 3: 7

Enter the size of the files:
File 1: 1
File 2: 4

File_no:        File_size:      Block_no:       Block_size:     Fragment
1               1               3               7               6
2               4               1               5               1
eshwar@eshwar-VirtualBox:~/Desktop$
```

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

int fr[3];

void display();

int main() {
          printf("\tPage Replacement Algorithm - FIFO");

   int i, j, page[12] = {2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2};
   int flag1 = 0, flag2 = 0, pf = 0, frsize = 3, top = 0;

   for(i = 0; i < 3; i++) {
      fr[i] = -1;
   }

   for(j = 0; j < 12; j++) {
      flag1 = 0;
      flag2 = 0;

      for(i = 0; i < 3; i++) {
         if(fr[i] == page[j]) {
            flag1 = 1;
            flag2 = 1;
            break;
         }
      }

      if(flag1 == 0) {
         for(i = 0; i < frsize; i++) {
            if(fr[i] == -1) {
               fr[i] = page[j];
               flag2 = 1;
               break;
            }
         }

         if(flag2 == 0) {
            fr[top] = page[j];
            top++;
            pf++;

            if(top >= frsize) {
                    top = 0;
                }

                display();
            }
        }
    }

    printf("\nNumber of page faults: %d\n", pf + frsize);
    return 0;
}

void display() {
    int i;

    printf("\n");

    for(i = 0; i < 3; i++) {
        printf("%d\t", fr[i]);
    }
}
```

Output:



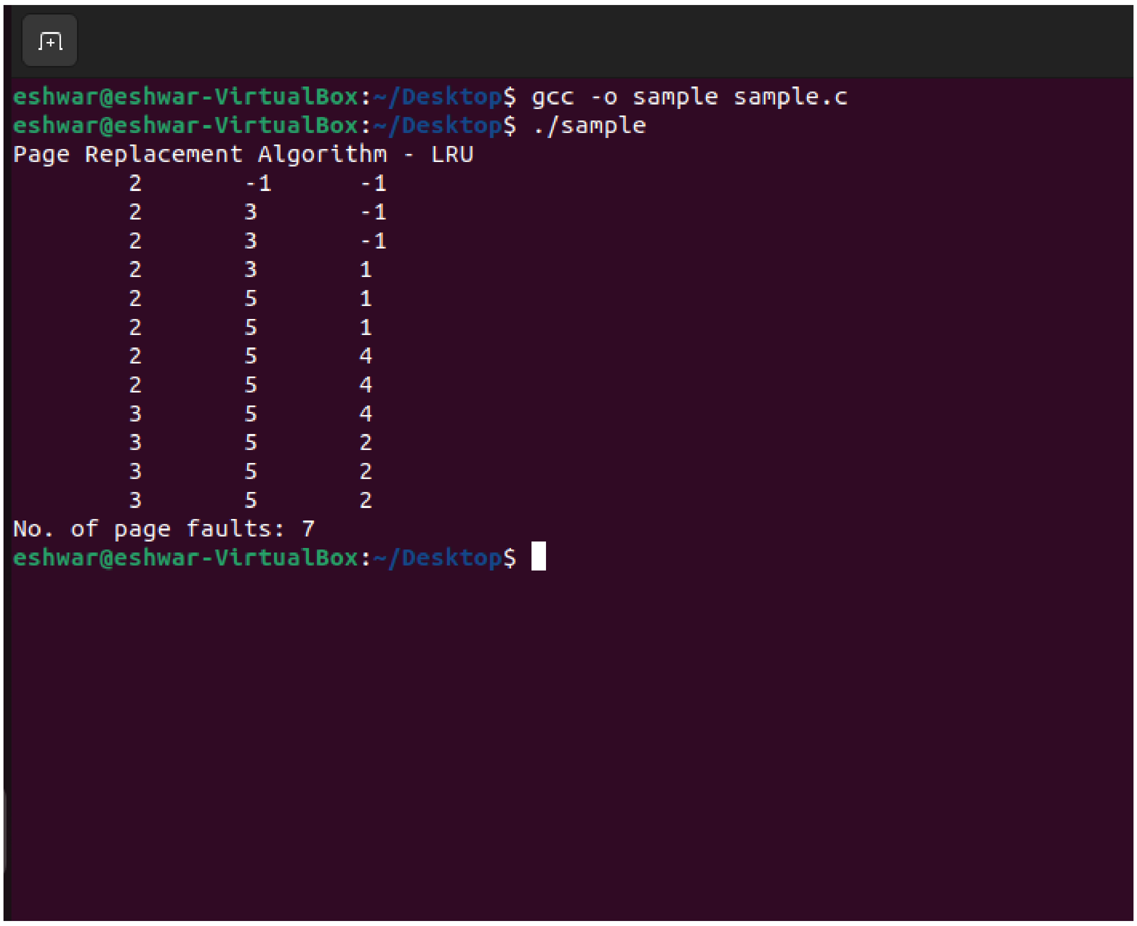
```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
        Page Replacement Algorithm - FIFO
5       3       1
5       2       1
5       2       4
3       2       4
3       5       4
3       5       2
Number of page faults: 9
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

```
#include <stdio.h>

int fr[3];

void display();

void main() {
   printf("Page Replacement Algorithm – LRU");
   int p[12] = {2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2};
   int i, j, fs[3], index, k, l, flag1 = 0, flag2 = 0, pf = 0, frsize = 3;

   for(i = 0; i < 3; i++)
      fr[i] = -1;

   for(j = 0; j < 12; j++) {
      flag1 = 0;
      flag2 = 0;

      for(i = 0; i < 3; i++) {
         if(fr[i] == p[j]) {
            flag1 = 1;
            flag2 = 1;
            break;
         }
      }

      if(flag1 == 0) {
         for(i = 0; i < 3; i++) {
            if(fr[i] == -1) {
               fr[i] = p[j];
               flag2 = 1;
               break;
            }
         }

         if(flag2 == 0) {
            for(i = 0; i < 3; i++)
               fs[i] = 0;

            for(k = j - 1, l = 1; l <= frsize - 1; l++, k--)
               for(i = 0; i < 3; i++)
                  if(fr[i] == p[k])
                     fs[i] = 1;
```

```
                for(i = 0; i < 3; i++) {
                    if(fs[i] == 0) {
                        index = i;
                        break;
                    }
                }

                fr[index] = p[j];
                pf++;
            }
        }

        display();
    }

    printf("\nNo. of page faults: %d\n", pf + frsize);
}

void display() {
    int i;
    printf("\n");

    for(i = 0; i < 3; i++)
        printf("\t%d", fr[i]);
}
```

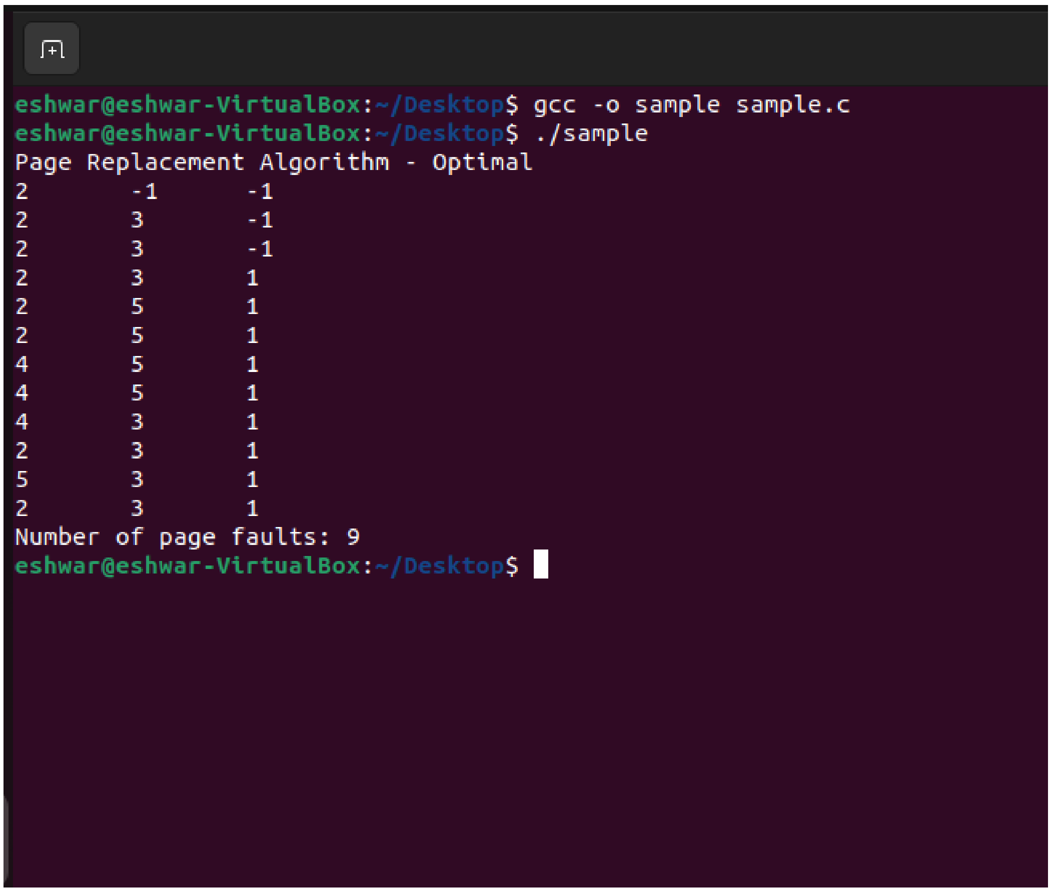
Output:



```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Page Replacement Algorithm - LRU
        2       -1      -1
        2       3       -1
        2       3       -1
        2       3       1
        2       5       1
        2       5       1
        2       5       4
        2       5       4
        3       5       4
        3       5       2
        3       5       2
        3       5       2
No. of page faults: 7
eshwar@eshwar-VirtualBox:~/Desktop$ 
```

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

int fr[3], n, m;

void display();

void main() {
    int i, j, page[12] = {2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2};
    int max, found = 0, lg[3], index, k, l, flag1 = 0, flag2 = 0, pf = 0;

    n = 12; // Length of the reference string
    m = 3; // Number of frames

    for(i = 0; i < 3; i++)
        fr[i] = -1;

    pf = m;

    printf("Page Replacement Algorithm - Optimal\n");

    for(j = 0; j < 12; j++) {
        flag1 = 0;
        flag2 = 0;

        for(i = 0; i < m; i++) {
            if(fr[i] == page[j]) {
                flag1 = 1;
                break;
            }
        }

        if(flag1 == 0) {
            for(i = 0; i < m; i++) {
                if(fr[i] == -1) {
                    fr[i] = page[j];
                    flag2 = 1;
                    break;
                }
            }

            if(flag2 == 0) {
                for(i = 0; i < m; i++)
                    lg[i] = 0;
                for(i = 0; i < m; i++) {
                    found = 0;
                    for(k = j + 1; k < n; k++) {
                        if(fr[i] == page[k]) {
                            lg[i] = k - j;
                            found = 1;
                            break;
                        }
                    }
                    if(found == 0) {
                        lg[i] = 0;
                    }
                }

                max = lg[0];
                index = 0;
                for(i = 0; i < m; i++) {
                    if(max < lg[i]) {
                        max = lg[i];
                        index = i;
                    }
                }

                fr[index] = page[j];
                pf++;
            }
        }

        display();
    }

    printf("Number of page faults: %d\n", pf);
}

void display() {
    int i;

    for(i = 0; i < m; i++)
        printf("%d\t", fr[i]);
    printf("\n");
}
```

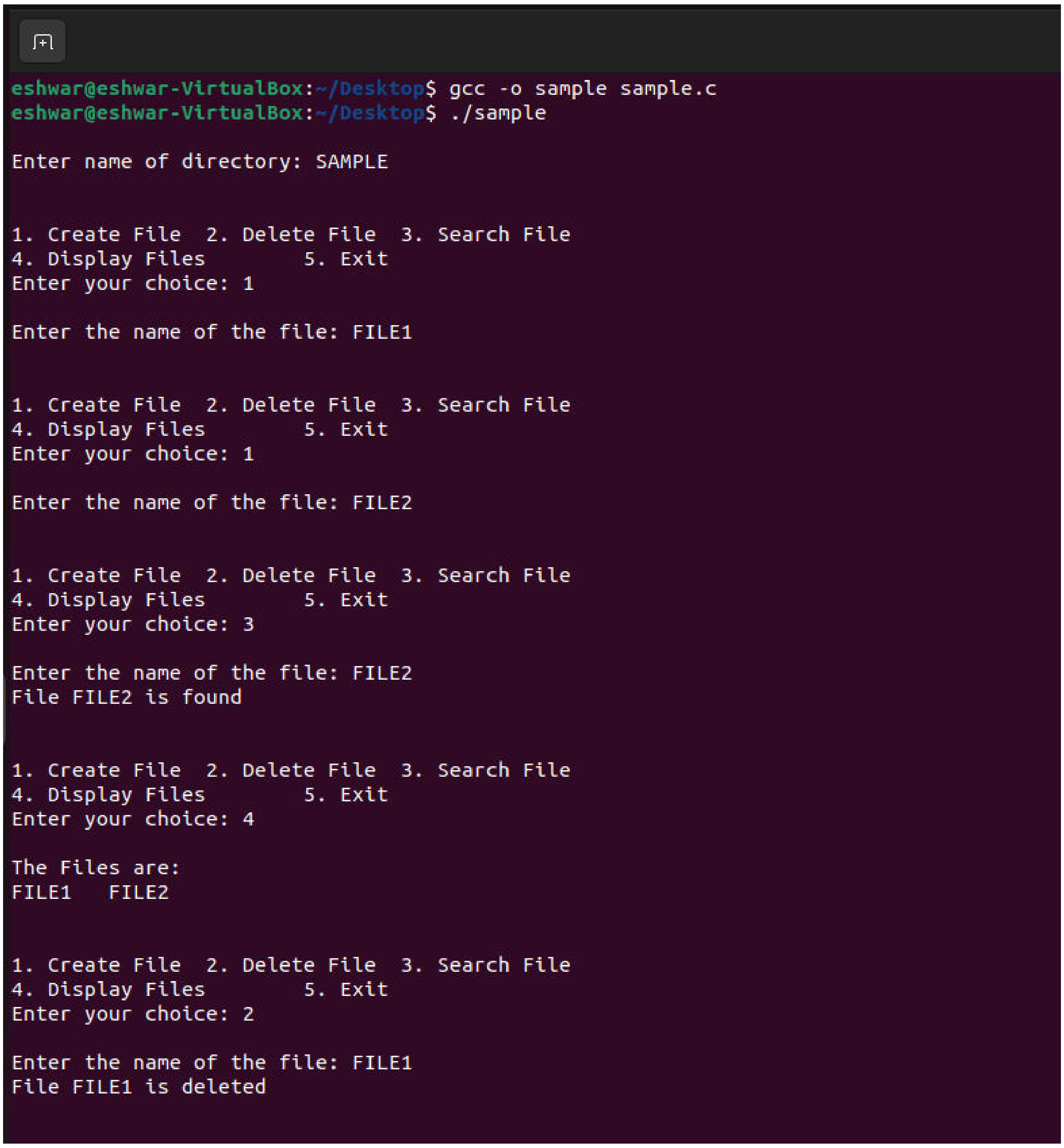
Output:



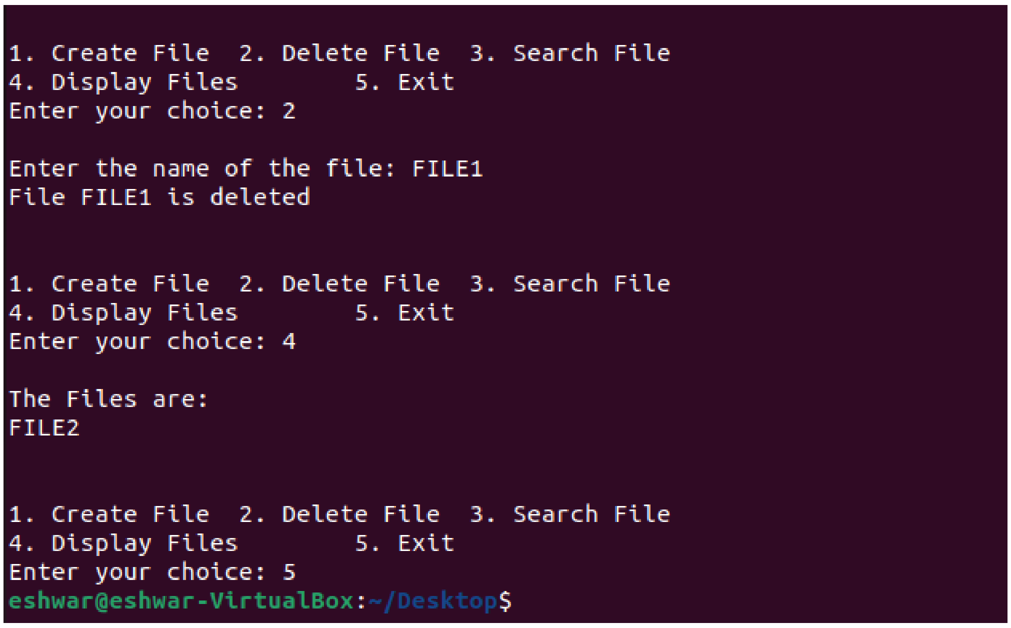
```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Page Replacement Algorithm - Optimal
2       -1      -1
2       3       -1
2       3       -1
2       3       1
2       5       1
2       5       1
4       5       1
4       5       1
4       3       1
2       3       1
5       3       1
2       3       1
Number of page faults: 9
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:
```
#include <stdio.h>
#include <string.h>

struct Directory {
   char dname[10];
   char fname[10][10];
   int fcnt;
} dir;

void main() {
   int i, ch;
   char f[30];
printf("File organization – Single level Directory");
   printf("\nEnter name of directory: ");
   scanf("%s", dir.dname);
   dir.fcnt = 0;

   while(1) {
      printf("\n\n1. Create File\t2. Delete File\t3. Search File\n4. Display Files\t5. Exit\nEnter
your choice: ");
      scanf("%d", &ch);

      switch(ch) {
         case 1:
            printf("\nEnter the name of the file: ");
            scanf("%s", dir.fname[dir.fcnt]);
            dir.fcnt++;
            break;

         case 2:
            printf("\nEnter the name of the file: ");
            scanf("%s", f);
            for(i = 0; i < dir.fcnt; i++) {
               if(strcmp(f, dir.fname[i]) == 0) {
                  printf("File %s is deleted\n", f);
                  strcpy(dir.fname[i], dir.fname[dir.fcnt - 1]);
                  dir.fcnt--;
                  break;
               }
            }
            if(i == dir.fcnt)
               printf("File %s not found\n", f);
            break;
            case 3:
                printf("\nEnter the name of the file: ");
                scanf("%s", f);
                for(i = 0; i < dir.fcnt; i++) {
                    if(strcmp(f, dir.fname[i]) == 0) {
                        printf("File %s is found\n", f);
                        break;
                    }
                }
                if(i == dir.fcnt)
                    printf("File %s not found\n", f);
                break;

            case 4:
                if(dir.fcnt == 0)
                    printf("\nDirectory is Empty\n");
                else {
                    printf("\nThe Files are:\n");
                    for(i = 0; i < dir.fcnt; i++)
                        printf("%s\t", dir.fname[i]);
                    printf("\n");
                }
                break;

            case 5:
                return;

            default:
                printf("\nInvalid choice\n");
        }
    }
}
```

Output:



```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample

Enter name of directory: SAMPLE


1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice: 1

Enter the name of the file: FILE1


1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice: 1

Enter the name of the file: FILE2


1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice: 3

Enter the name of the file: FILE2
File FILE2 is found


1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice: 4

The Files are:
FILE1   FILE2


1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice: 2

Enter the name of the file: FILE1
File FILE1 is deleted
```



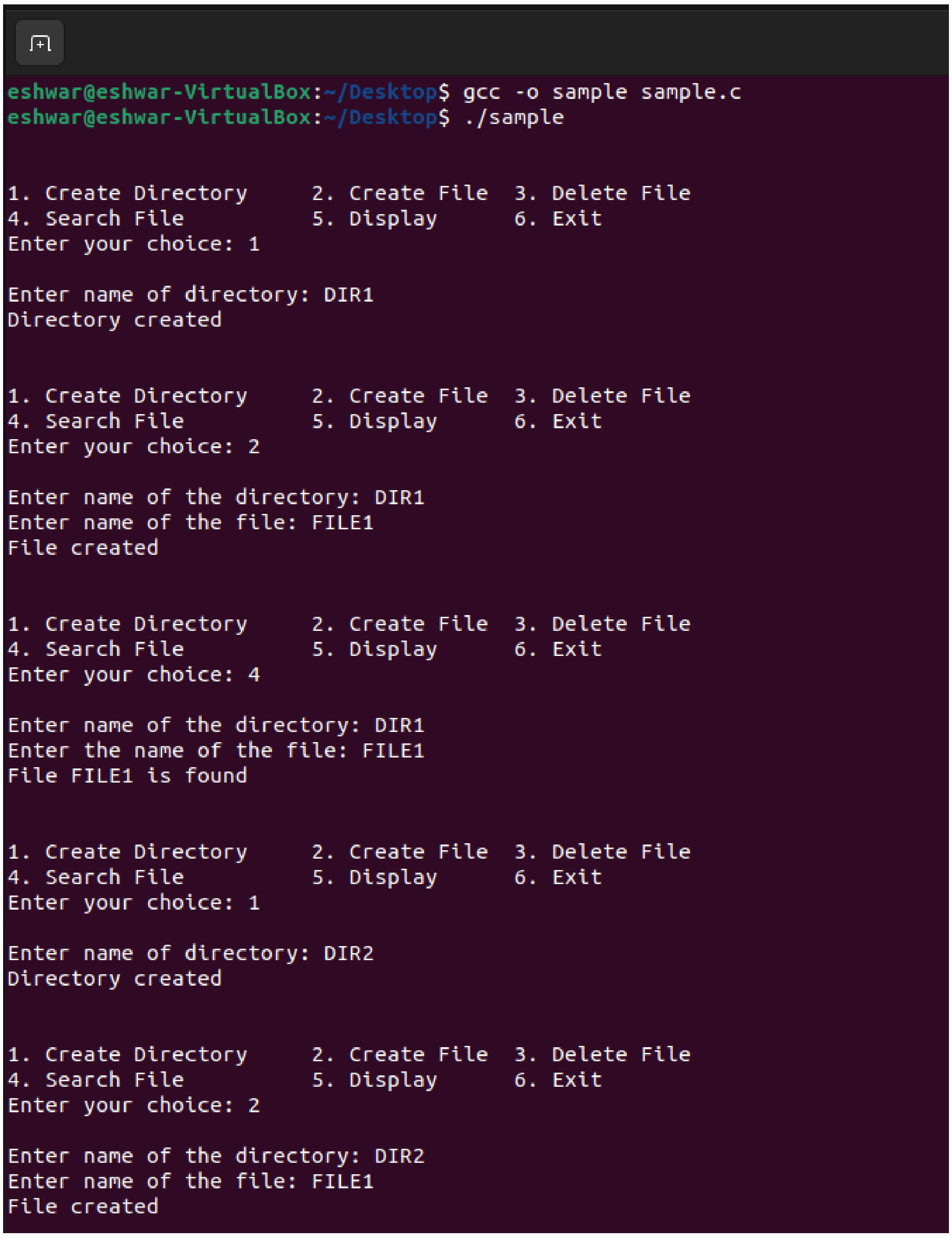
```
1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice: 2

Enter the name of the file: FILE1
File FILE1 is deleted


1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice: 4

The Files are:
FILE2


1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice: 5
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

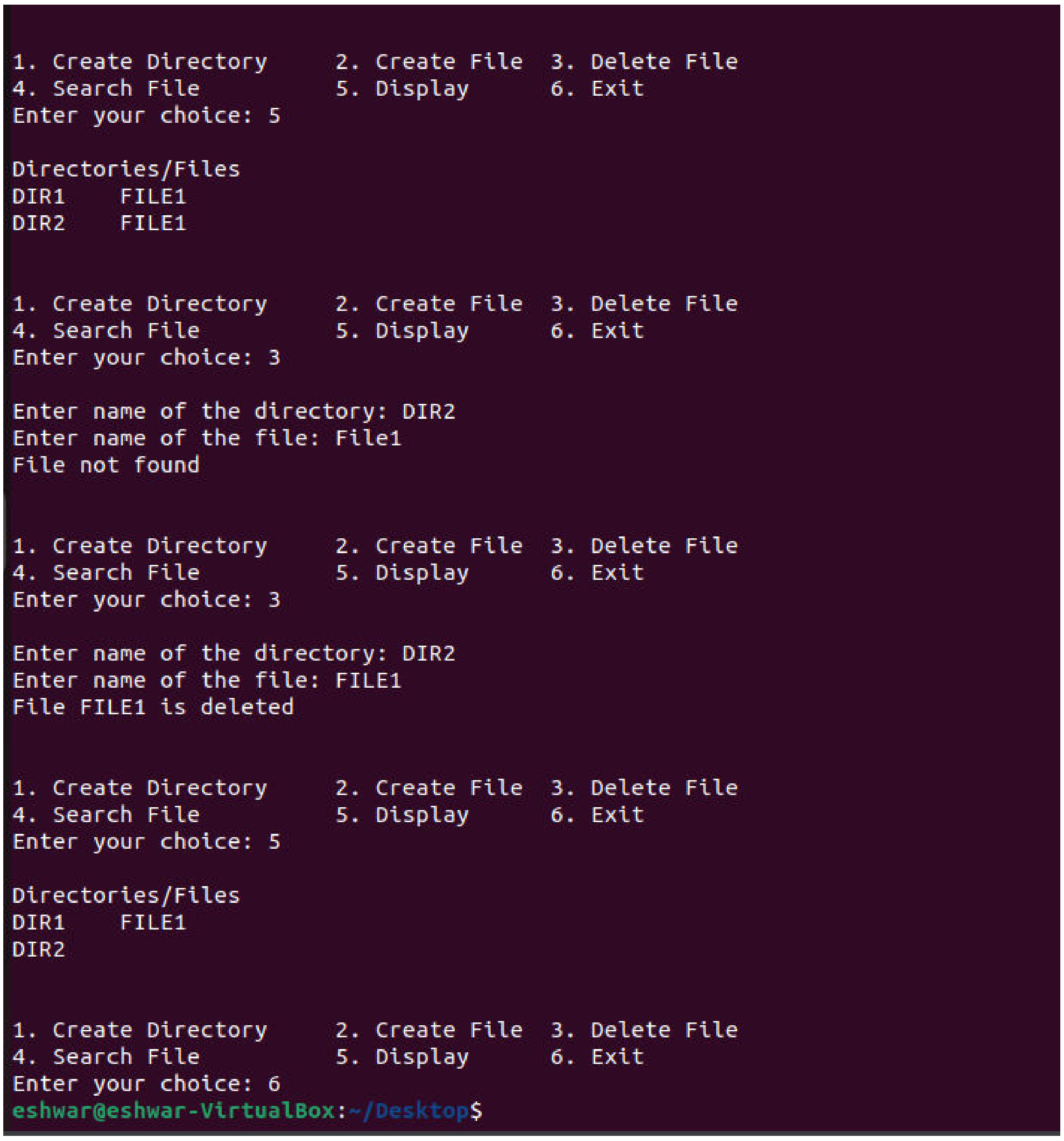
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct Directory {
   char dname[10];
   char fname[10][10];
   int fcnt;
} dir[10];

void main() {
   int I, ch, dent, dcnt = 0, k;
   char f[30], d[30];
printf("File organization – Two level Directory");

   while(1) {
      printf("\n\n1. Create Directory\t2. Create File\t3. Delete File");
      printf("\n4. Search File\t\t5. Display\t6. Exit\nEnter your choice: ");
      scanf("%d", &ch);

      switch(ch) {
         case 1:
            printf("\nEnter name of directory: ");
            scanf("%s", dir[dcnt].dname);
            dir[dcnt].fcnt = 0;
            dcnt++;
            printf("Directory created\n");
            break;

         case 2:
            printf("\nEnter name of the directory: ");
            scanf("%s", d);

            for(I = 0; I < dcnt; i++) {
               if(strcmp(d, dir[i].dname) == 0) {
                  printf("Enter name of the file: ");
                  scanf("%s", dir[i].fname[dir[i].fcnt]);
                  dir[i].fcnt++;
                  printf("File created\n");
                  break;
               }
            }
```

```
    if(I == dcnt)
        printf(“Directory not found\n”);

    break;

case 3:
    printf(“\nEnter name of the directory: “);
    scanf(“%s”, d);

    for(I = 0; I < dcnt; i++) {
        if(strcmp(d, dir[i].dname) == 0) {
            printf(“Enter name of the file: “);
            scanf(“%s”, f);

            for(k = 0; k < dir[i].fcnt; k++) {
                if(strcmp(f, dir[i].fname[k]) == 0) {
                    printf(“File %s is deleted\n”, f);
                    dir[i].fcnt--;
                    strcpy(dir[i].fname[k], dir[i].fname[dir[i].fcnt]);
                    goto jmp;
                }
            }

            printf(“File not found\n”);
            goto jmp;
        }
    }

    printf(“Directory not found\n”);
    jmp:
    break;

case 4:
    printf(“\nEnter name of the directory: “);
    scanf(“%s”, d);

    for(I = 0; I < dcnt; i++) {
        if(strcmp(d, dir[i].dname) == 0) {
            printf(“Enter the name of the file: “);
            scanf(“%s”, f);

            for(k = 0; k < dir[i].fcnt; k++) {
                if(strcmp(f, dir[i].fname[k]) == 0) {
                    printf(“File %s is found\n”, f);
                    goto jmp1;
                  }
                }

                printf("File not found\n");
                goto jmp1;
              }
            }

            printf("Directory not found\n");
            jmp1:
            break;

          case 5:
            if(dcnt == 0)
              printf("\nNo Directories\n");
            else {
              printf("\nDirectories/Files\n");

              for(I = 0; I < dcnt; i++) {
                printf("%s\t", dir[i].dname);

                for(k = 0; k < dir[i].fcnt; k++)
                  printf("%s\t", dir[i].fname[k]);

                printf("\n");
              }
            }
            break;

          case 6:
            exit(0);
            break;

          default:
            printf("\nInvalid choice\n");
        }
      }
    }
```

Output:



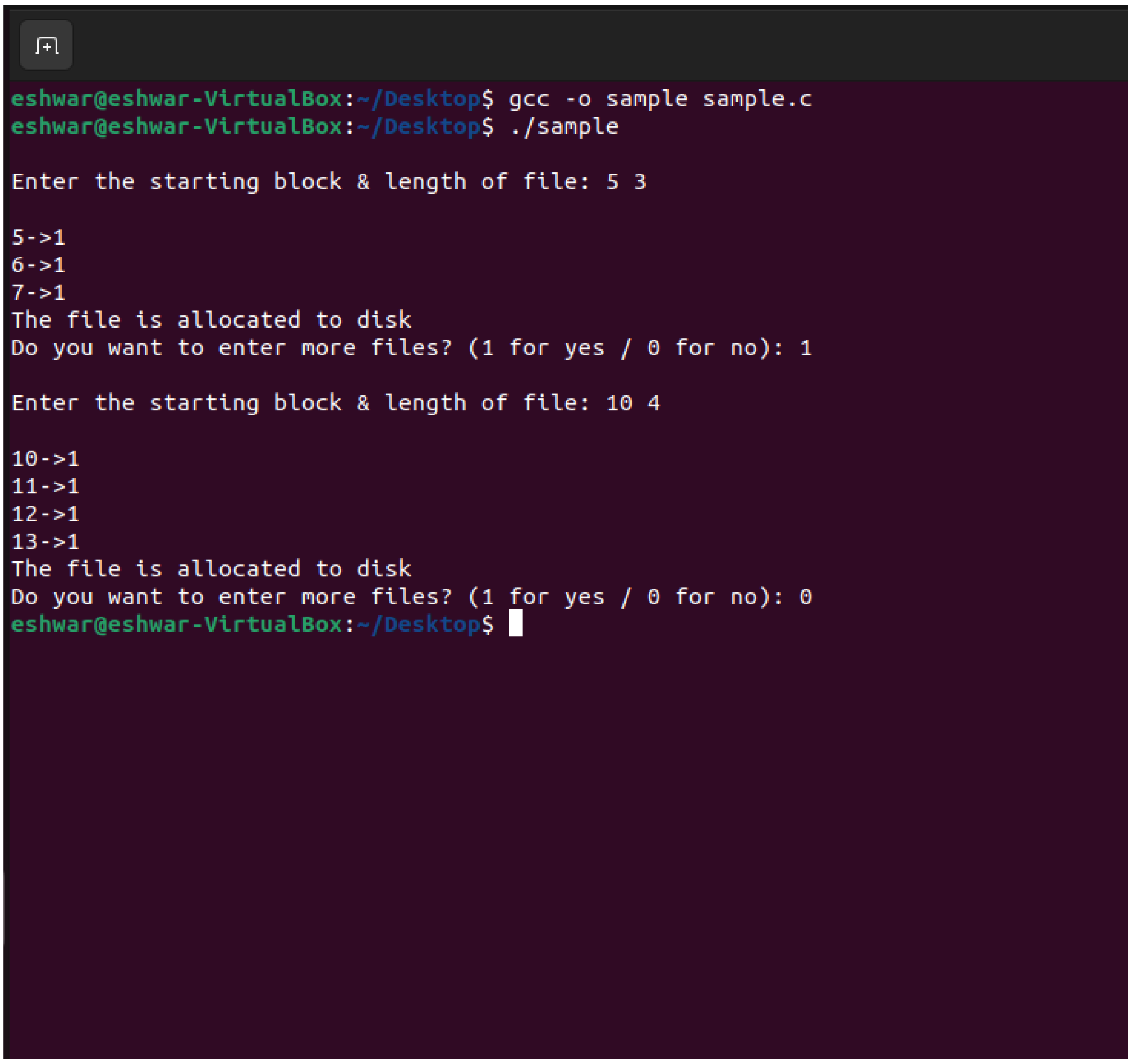
```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 1

Enter name of directory: DIR1
Directory created


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 2

Enter name of the directory: DIR1
Enter name of the file: FILE1
File created


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 4

Enter name of the directory: DIR1
Enter the name of the file: FILE1
File FILE1 is found


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 1

Enter name of directory: DIR2
Directory created


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 2

Enter name of the directory: DIR2
Enter name of the file: FILE1
File created
```



```
1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 5

Directories/Files
DIR1    FILE1
DIR2    FILE1


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 3

Enter name of the directory: DIR2
Enter name of the file: File1
File not found


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 3

Enter name of the directory: DIR2
Enter name of the file: FILE1
File FILE1 is deleted


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 5

Directories/Files
DIR1    FILE1
DIR2


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice: 6
eshwar@eshwar-VirtualBox:~/Desktop$
```

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

void main() {
   int f[50], i, st, j, len, c, k;

   for(i = 0; i < 50; i++)
      f[i] = 0;

printf(“File Allocation Strategies - Sequential”);

   X:
   printf("\nEnter the starting block & length of file: ");
   scanf("%d%d", &st, &len);

   for(j = st; j < (st + len); j++) {
      if(f[j] == 0) {
         f[j] = 1;
         printf("\n%d->%d", j, f[j]);
      } else {
         printf("\nBlock already allocated");
         break;
      }
   }

   if(j == (st + len))
      printf("\nThe file is allocated to disk");

   printf("\nDo you want to enter more files? (1 for yes / 0 for no): ");
   scanf("%d", &c);

   if(c == 1)
      goto X;
   else
      exit(0);

}
```

Output:

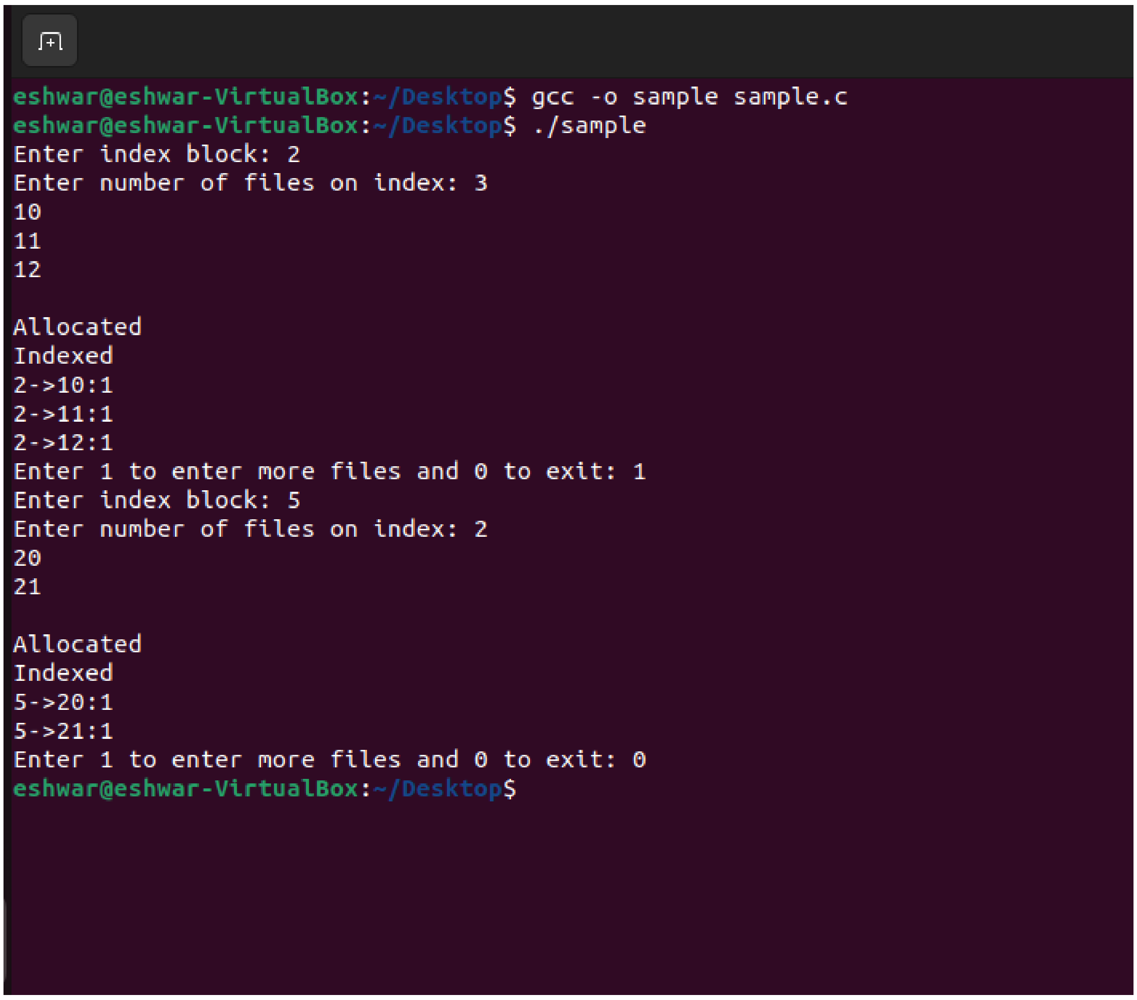


```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample

Enter the starting block & length of file: 5 3

5->1
6->1
7->1
The file is allocated to disk
Do you want to enter more files? (1 for yes / 0 for no): 1

Enter the starting block & length of file: 10 4

10->1
11->1
12->1
13->1
The file is allocated to disk
Do you want to enter more files? (1 for yes / 0 for no): 0
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

```
#include<stdio.h>
#include<stdlib.h>
int f[50], i, k, j, inde[50], n, c, count = 0, p;

void main() {
   for(i = 0; i < 50; i++)
      f[i] = 0;
printf(“File Allocation Strategies - Indexed”);
   x:
   printf("Enter index block: ");
   scanf("%d", &p);

   if(f[p] == 0) {
      f[p] = 1;
      printf("Enter number of files on index: ");
      scanf("%d", &n);
   } else {
      printf("Block already allocated\n");
      goto x;
   }
   for(i = 0; i < n; i++)
      scanf("%d", &inde[i]);
   for(i = 0; i < n; i++) {
      if(f[inde[i]] == 1) {
         printf("Block already allocated");
         goto x;
      }
   }

   for(j = 0; j < n; j++)
      f[inde[j]] = 1;

   printf("\nAllocated\nIndexed\n");
   for(k = 0; k < n; k++)
      printf("%d->%d:%d\n", p, inde[k], f[inde[k]]);

   printf("Enter 1 to enter more files and 0 to exit: ");
   scanf("%d", &c);
   if(c == 1)
      goto x;
   else
      exit(0);
}
```

Output:

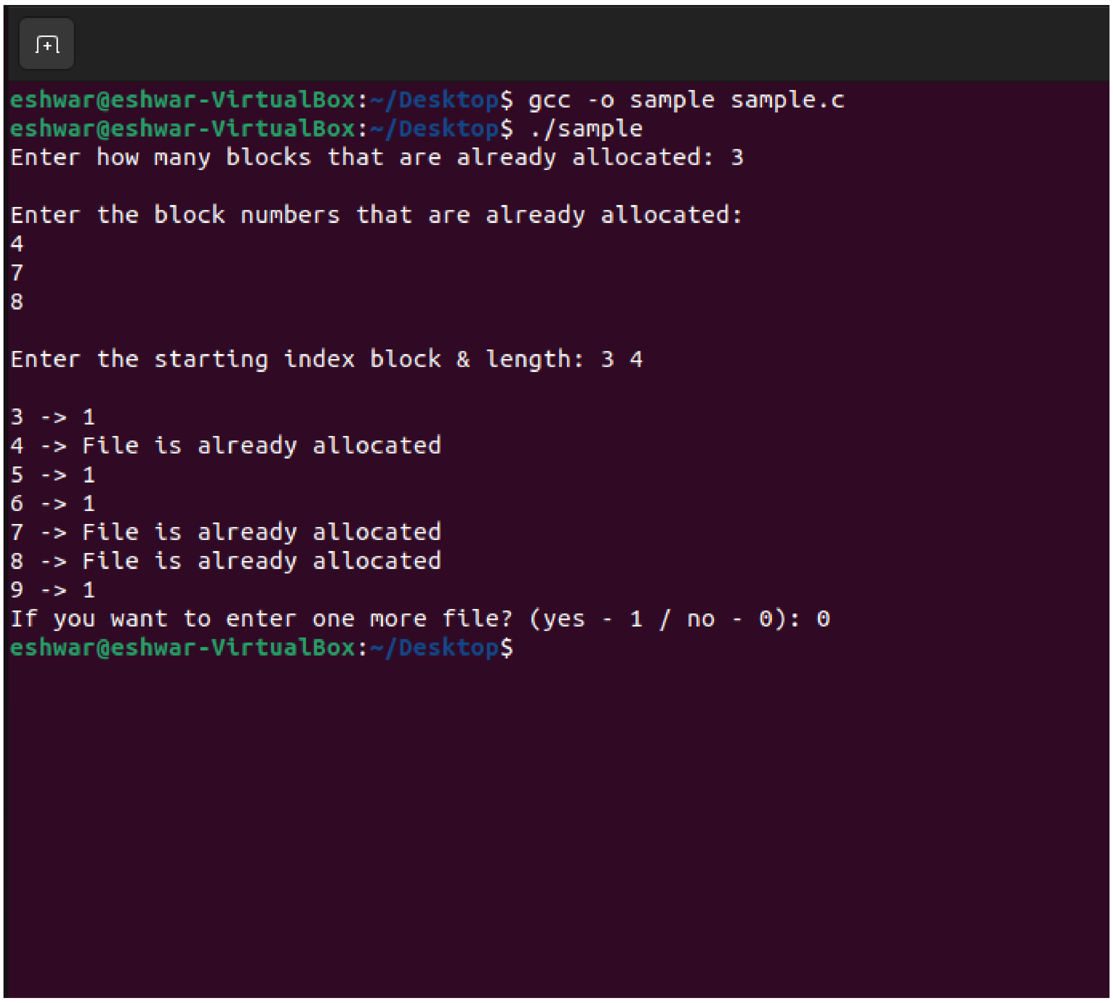


```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Enter index block: 2
Enter number of files on index: 3
10
11
12

Allocated
Indexed
2->10:1
2->11:1
2->12:1
Enter 1 to enter more files and 0 to exit: 1
Enter index block: 5
Enter number of files on index: 2
20
21

Allocated
Indexed
5->20:1
5->21:1
Enter 1 to enter more files and 0 to exit: 0
eshwar@eshwar-VirtualBox:~/Desktop$
```

Program:

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

void main() {
   int f[50], p, i, j, k, st, len, n, c;
printf("File Allocation Strategies - Linked");
   for(i = 0; i < 50; i++)
      f[i] = 0;

   printf("Enter how many blocks that are already allocated: ");
   scanf("%d", &p);

   printf("\nEnter the block numbers that are already allocated:\n");
   for(i = 0; i < p; i++) {
      scanf("%d", &n);
      f[n] = 1;
   }

   X:
   printf("\nEnter the starting index block & length: ");
   scanf("%d%d", &st, &len);

   k = len;
   for(j = st; j < (k + st); j++) {
      if(f[j] == 0) {
         f[j] = 1;
         printf("\n%d -> %d ", j, f[j]);
      } else {
         printf("\n%d -> File is already allocated", j);
         k++;
      }
   }

   printf("\nIf you want to enter one more file? (yes - 1 / no - 0): ");
   scanf("%d", &c);
   if(c == 1)
      goto X;
   else
      exit(0);

}
```

Output:



```
eshwar@eshwar-VirtualBox:~/Desktop$ gcc -o sample sample.c
eshwar@eshwar-VirtualBox:~/Desktop$ ./sample
Enter how many blocks that are already allocated: 3

Enter the block numbers that are already allocated:
4
7
8

Enter the starting index block & length: 3 4

3 -> 1
4 -> File is already allocated
5 -> 1
6 -> 1
7 -> File is already allocated
8 -> File is already allocated
9 -> 1
If you want to enter one more file? (yes - 1 / no - 0): 0
eshwar@eshwar-VirtualBox:~/Desktop$
```