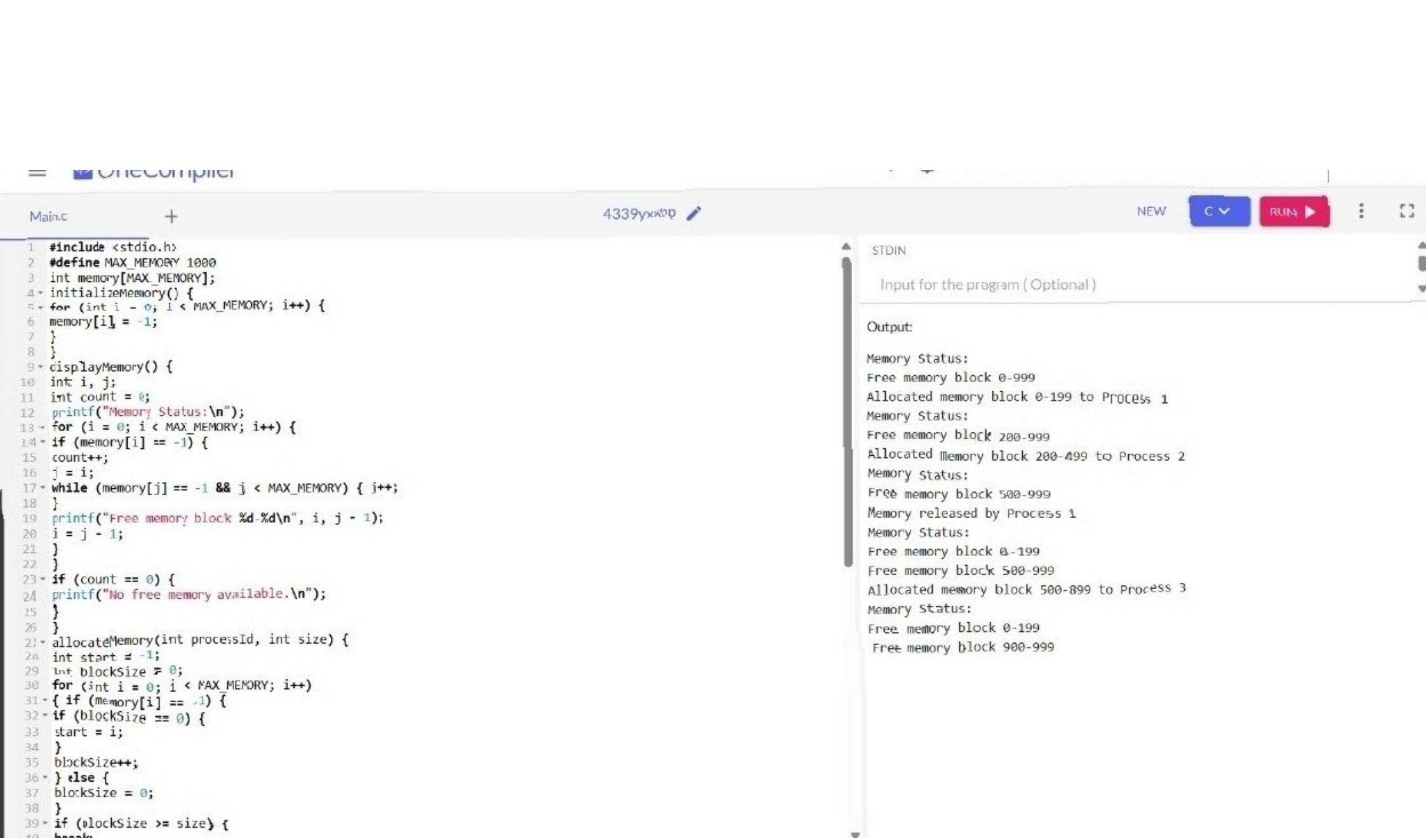
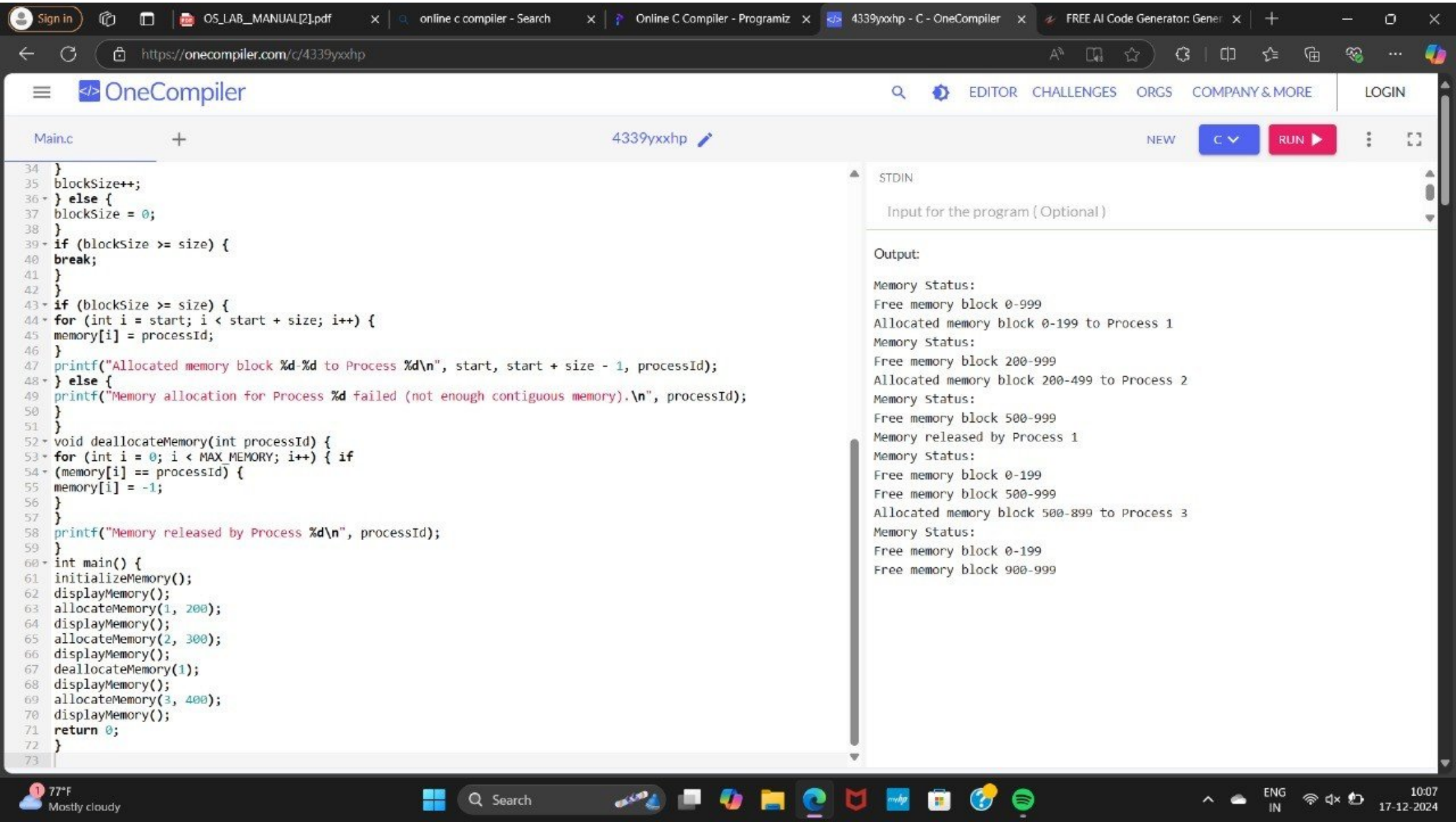
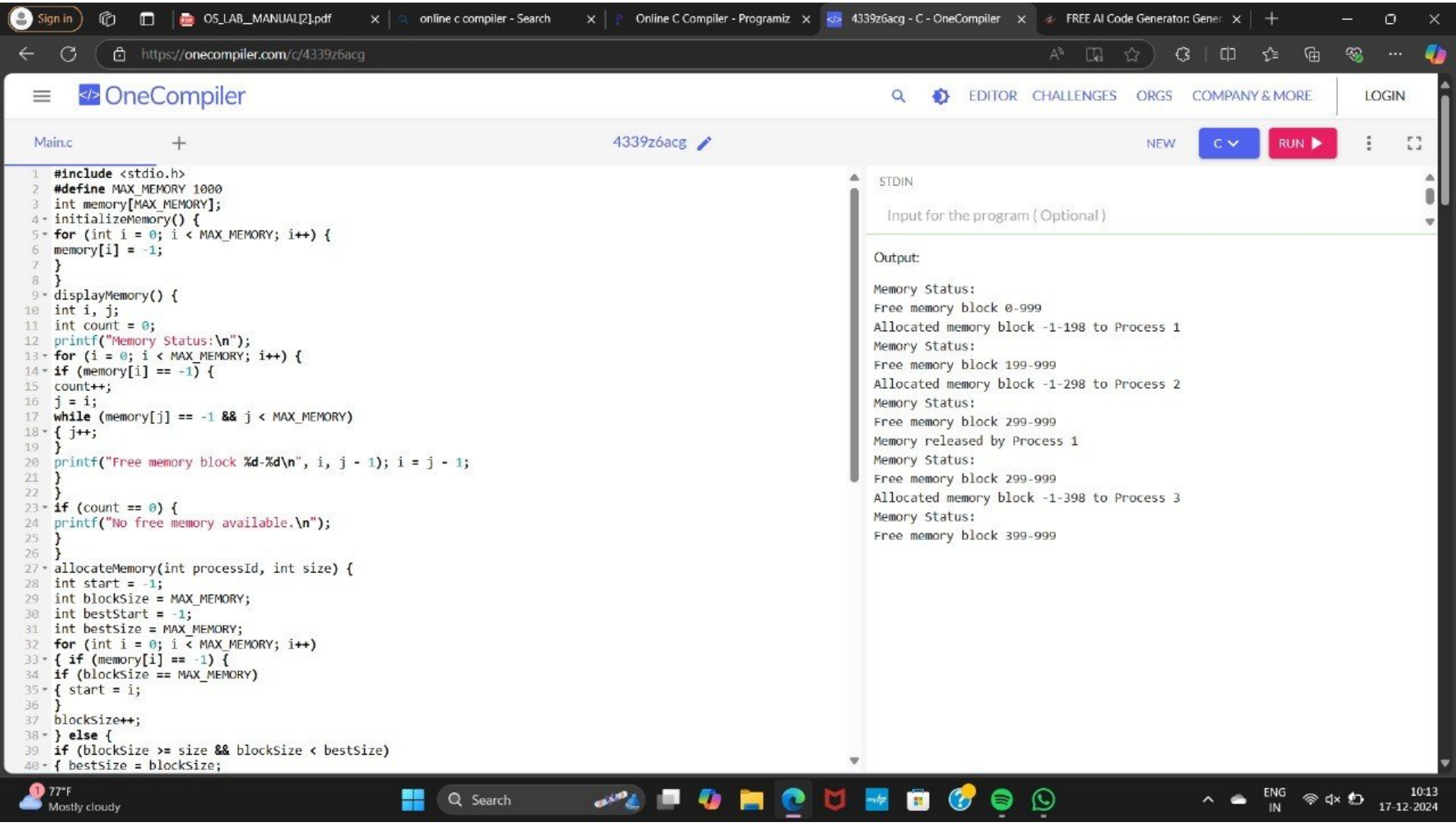
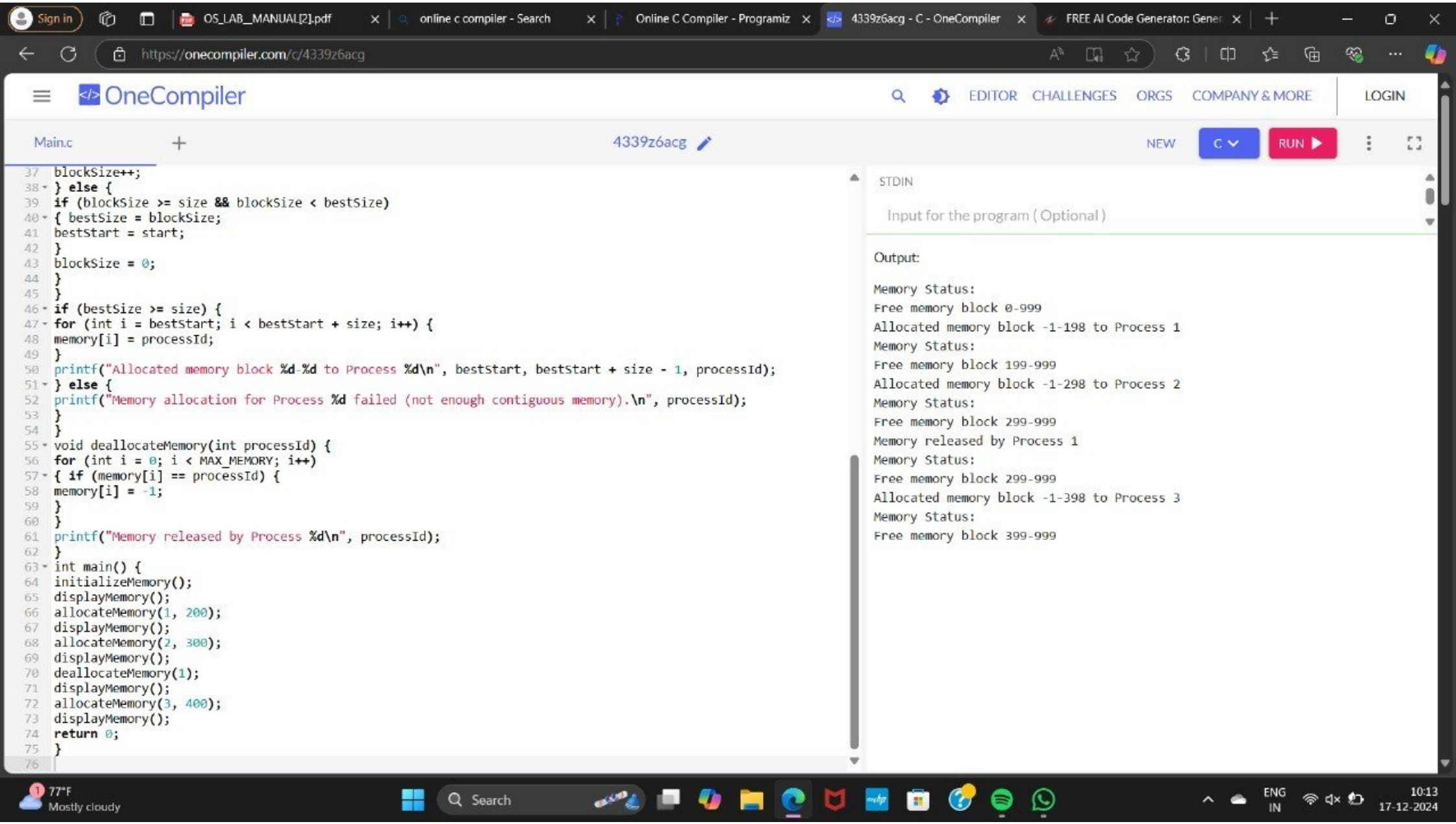
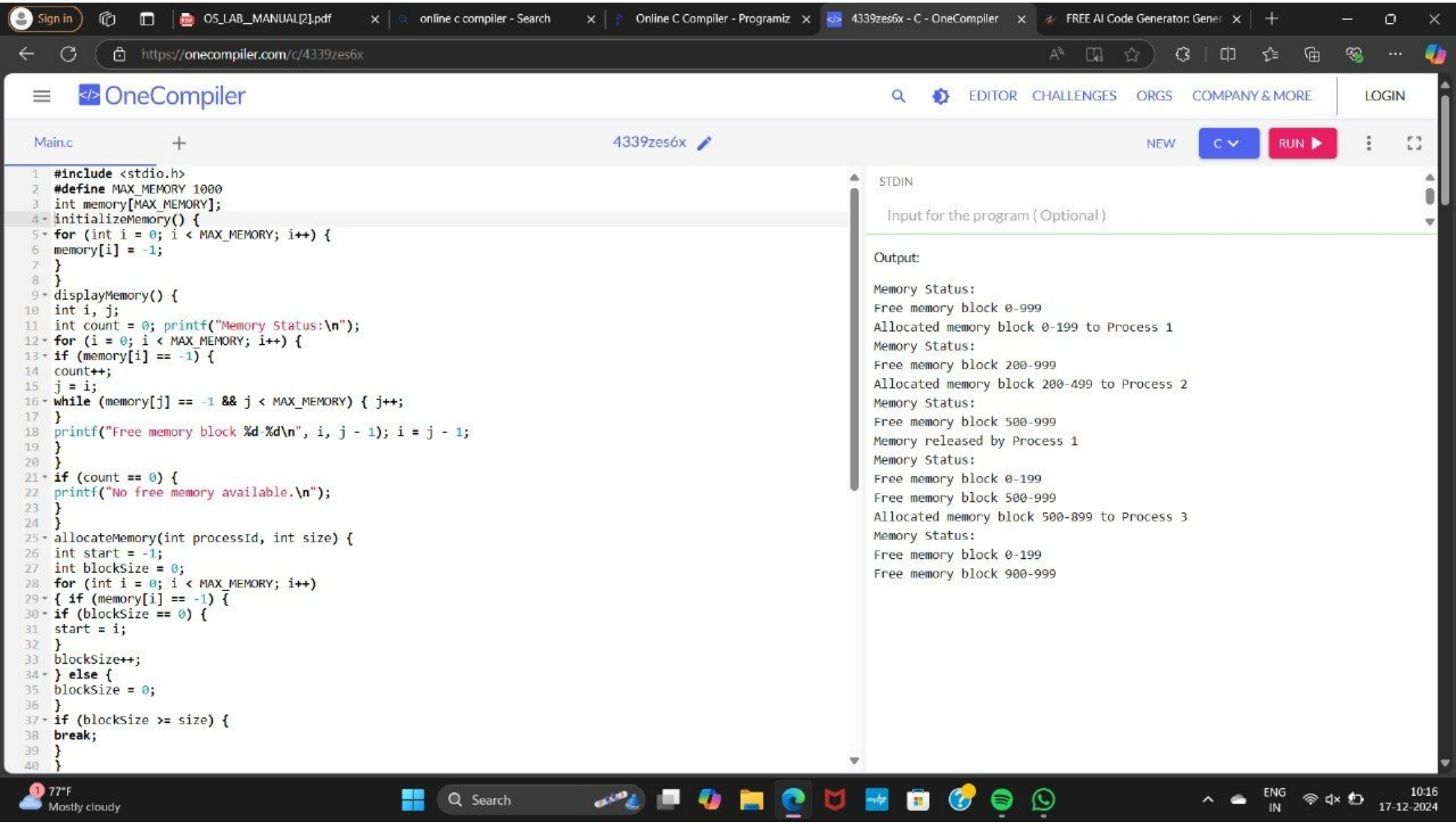


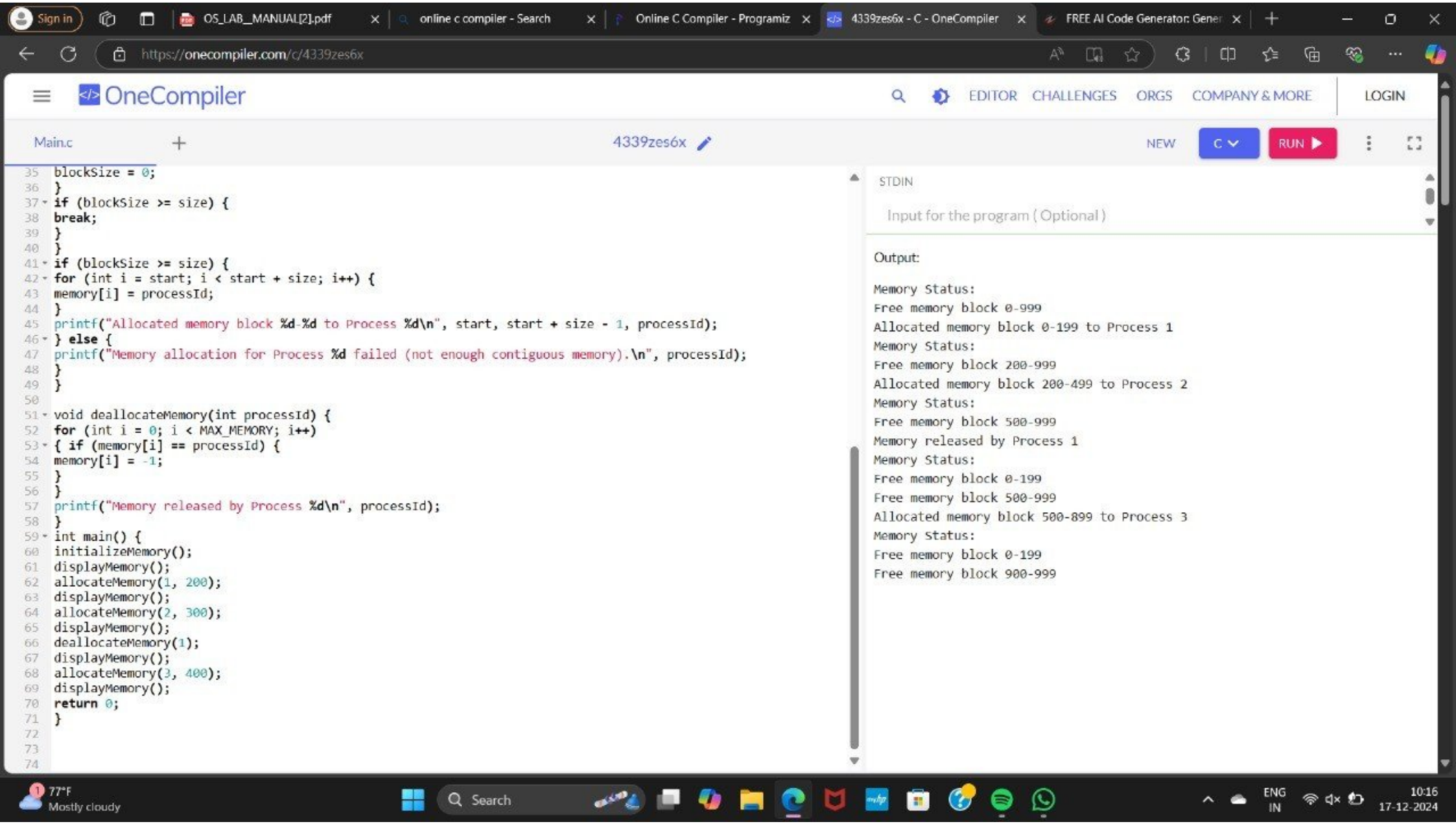

```c
#include <stdio.h>
#define MAX_MEMORY 1000
int memory[MAX_MEMORY];
initializeMemory() {
for (int i = 0; i < MAX_MEMORY; i++) {
memory[i] = -1;
}
}
displayMemory() {
int i, j;
int count = 0;
printf("Memory Status:\n");
for (i = 0; i < MAX_MEMORY; i++) {
if (memory[i] == -1) {
count++;
j = i;
while (memory[j] == -1 && j < MAX_MEMORY) { j++;
}
printf("Free memory block %d-%d\n", i, j - 1);
i = j - 1;
}
}
if (count == 0) {
printf("No free memory available.\n");
}
}
allocateMemory(int processId, int size) {
int start = -1;
int blockSize = 0;
for (int i = 0; i < MAX_MEMORY; i++)
{ if (memory[i] == -1) {
if (blockSize == 0) {
start = i;
}
blockSize++;
} else {
blockSize = 0;
}
if (blockSize >= size) {
```


Sign in
OS_LAB_MANUAL[2].pdf
online c compiler - Search
Online C Compiler - Programiz
4339yxxhp - C - OneCompiler
FREE AI Code Generator: Gener
https://onecompiler.com/c/4339yxxhp
OneCompiler
EDITOR
CHALLENGES
ORGS
COMPANY & MORE
LOGIN
Main.c
4339yxxhp
NEW
C
RUN
}
blockSize++;
} else {
blockSize = 0;
}
if (blockSize >= size) {
break;
}
}
if (blockSize >= size) {
for (int i = start; i < start + size; i++) {
memory[i] = processId;
}
printf("Allocated memory block %d-%d to Process %d\n", start, start + size - 1, processId);
} else {
printf("Memory allocation for Process %d failed (not enough contiguous memory).\n", processId);
}
}
void deallocateMemory(int processId) {
for (int i = 0; i < MAX_MEMORY; i++) { if
(memory[i] == processId) {
memory[i] = -1;
}
}
printf("Memory released by Process %d\n", processId);
}
int main() {
initializeMemory();
displayMemory();
allocateMemory(1, 200);
displayMemory();
allocateMemory(2, 300);
displayMemory();
deallocateMemory(1);
displayMemory();
allocateMemory(3, 400);
displayMemory();
return 0;
}
STDIN
Input for the program (Optional)
Output:
Memory Status:
Free memory block 0-999
Allocated memory block 0-199 to Process 1
Memory Status:
Free memory block 200-999
Allocated memory block 200-499 to Process 2
Memory Status:
Free memory block 500-999
Memory released by Process 1
Memory Status:
Free memory block 0-199
Free memory block 500-999
Allocated memory block 500-899 to Process 3
Memory Status:
Free memory block 0-199
Free memory block 900-999
77°F
Mostly cloudy
Search
ENG
IN
10:07
17-12-2024


Sign in
OS_LAB_MANUAL[2].pdf
online c compiler - Search
Online C Compiler - Programiz
4339z6acg - C - OneCompiler
FREE AI Code Generator: Gener
https://onecompiler.com/c/4339z6acg
OneCompiler
EDITOR CHALLENGES ORGS COMPANY & MORE
LOGIN
Main.c
4339z6acg
NEW
C
RUN
#include <stdio.h>
#define MAX_MEMORY 1000
int memory[MAX_MEMORY];
initializeMemory() {
for (int i = 0; i < MAX_MEMORY; i++) {
memory[i] = -1;
}
}
displayMemory() {
int i, j;
int count = 0;
printf("Memory Status:\n");
for (i = 0; i < MAX_MEMORY; i++) {
if (memory[i] == -1) {
count++;
j = i;
while (memory[j] == -1 && j < MAX_MEMORY)
{ j++;
}
printf("Free memory block %d-%d\n", i, j - 1); i = j - 1;
}
}
if (count == 0) {
printf("No free memory available.\n");
}
}
allocateMemory(int processId, int size) {
int start = -1;
int blockSize = MAX_MEMORY;
int bestStart = -1;
int bestSize = MAX_MEMORY;
for (int i = 0; i < MAX_MEMORY; i++)
{ if (memory[i] == -1) {
if (blockSize == MAX_MEMORY)
{ start = i;
}
blockSize++;
} else {
if (blockSize >= size && blockSize < bestSize)
{ bestSize = blockSize;
STDIN
Input for the program (Optional)
Output:
Memory Status:
Free memory block 0-999
Allocated memory block -1-198 to Process 1
Memory Status:
Free memory block 199-999
Allocated memory block -1-298 to Process 2
Memory Status:
Free memory block 299-999
Memory released by Process 1
Memory Status:
Free memory block 299-999
Allocated memory block -1-398 to Process 3
Memory Status:
Free memory block 399-999
77°F
Mostly cloudy
Search
ENG
IN
10:13
17-12-2024


Sign in
OS_LAB_MANUAL[2].pdf
online c compiler - Search
Online C Compiler - Programiz
4339z6acg - C - OneCompiler
FREE AI Code Generator: Gener
https://onecompiler.com/c/4339z6acg
OneCompiler
EDITOR
CHALLENGES
ORGS
COMPANY & MORE
LOGIN
Main.c
4339z6acg
NEW
C
RUN
blockSize++;
} else {
if (blockSize >= size && blockSize < bestSize)
{ bestSize = blockSize;
bestStart = start;
}
blockSize = 0;
}
}
if (bestSize >= size) {
for (int i = bestStart; i < bestStart + size; i++) {
memory[i] = processId;
}
printf("Allocated memory block %d-%d to Process %d\n", bestStart, bestStart + size - 1, processId);
} else {
printf("Memory allocation for Process %d failed (not enough contiguous memory).\n", processId);
}
}
void deallocateMemory(int processId) {
for (int i = 0; i < MAX_MEMORY; i++)
{ if (memory[i] == processId) {
memory[i] = -1;
}
}
printf("Memory released by Process %d\n", processId);
}
int main() {
initializeMemory();
displayMemory();
allocateMemory(1, 200);
displayMemory();
allocateMemory(2, 300);
displayMemory();
deallocateMemory(1);
displayMemory();
allocateMemory(3, 400);
displayMemory();
return 0;
}
STDIN
Input for the program (Optional)
Output:
Memory Status:
Free memory block 0-999
Allocated memory block -1-198 to Process 1
Memory Status:
Free memory block 199-999
Allocated memory block -1-298 to Process 2
Memory Status:
Free memory block 299-999
Memory released by Process 1
Memory Status:
Free memory block 299-999
Allocated memory block -1-398 to Process 3
Memory Status:
Free memory block 399-999
77°F
Mostly cloudy
Search
ENG
IN
10:13
17-12-2024


Sign in
OS_LAB_MANUAL[2].pdf
online c compiler - Search
Online C Compiler - Programiz
4339zes6x - C - OneCompiler
FREE AI Code Generator: Gener
https://onecompiler.com/c/4339zes6x
OneCompiler
EDITOR CHALLENGES ORGS COMPANY & MORE
LOGIN
Main.c
4339zes6x
NEW
C
RUN
#include <stdio.h>
#define MAX_MEMORY 1000
int memory[MAX_MEMORY];
initializeMemory() {
for (int i = 0; i < MAX_MEMORY; i++) {
memory[i] = -1;
}
}
displayMemory() {
int i, j;
int count = 0; printf("Memory Status:\n");
for (i = 0; i < MAX_MEMORY; i++) {
if (memory[i] == -1) {
count++;
j = i;
while (memory[j] == -1 && j < MAX_MEMORY) { j++;
}
printf("Free memory block %d-%d\n", i, j - 1); i = j - 1;
}
}
if (count == 0) {
printf("No free memory available.\n");
}
}
allocateMemory(int processId, int size) {
int start = -1;
int blockSize = 0;
for (int i = 0; i < MAX_MEMORY; i++)
{ if (memory[i] == -1) {
if (blockSize == 0) {
start = i;
}
blockSize++;
} else {
blockSize = 0;
}
if (blockSize >= size) {
break;
}
}
STDIN
Input for the program (Optional)
Output:
Memory Status:
Free memory block 0-999
Allocated memory block 0-199 to Process 1
Memory Status:
Free memory block 200-999
Allocated memory block 200-499 to Process 2
Memory Status:
Free memory block 500-999
Memory released by Process 1
Memory Status:
Free memory block 0-199
Free memory block 500-999
Allocated memory block 500-899 to Process 3
Memory Status:
Free memory block 0-199
Free memory block 900-999
77°F
Mostly cloudy
Search
ENG
IN
10:16
17-12-2024


Sign in
OS_LAB_MANUAL[2].pdf
online c compiler - Search
Online C Compiler - Programiz
4339zes6x - C - OneCompiler
FREE AI Code Generator: Gener
https://onecompiler.com/c/4339zes6x
OneCompiler
EDITOR CHALLENGES ORGS COMPANY & MORE
LOGIN
Main.c
4339zes6x
NEW
C
RUN
blockSize = 0;
}
if (blockSize >= size) {
break;
}
}
if (blockSize >= size) {
for (int i = start; i < start + size; i++) {
memory[i] = processId;
}
printf("Allocated memory block %d-%d to Process %d\n", start, start + size - 1, processId);
} else {
printf("Memory allocation for Process %d failed (not enough contiguous memory).\n", processId);
}
}
void deallocateMemory(int processId) {
for (int i = 0; i < MAX_MEMORY; i++)
{ if (memory[i] == processId) {
memory[i] = -1;
}
}
printf("Memory released by Process %d\n", processId);
}
int main() {
initializeMemory();
displayMemory();
allocateMemory(1, 200);
displayMemory();
allocateMemory(2, 300);
displayMemory();
deallocateMemory(1);
displayMemory();
allocateMemory(3, 400);
displayMemory();
return 0;
}
STDIN
Input for the program (Optional)
Output:
Memory Status:
Free memory block 0-999
Allocated memory block 0-199 to Process 1
Memory Status:
Free memory block 200-999
Allocated memory block 200-499 to Process 2
Memory Status:
Free memory block 500-999
Memory released by Process 1
Memory Status:
Free memory block 0-199
Free memory block 500-999
Allocated memory block 500-899 to Process 3
Memory Status:
Free memory block 0-199
Free memory block 900-999
77°F
Mostly cloudy
Search
ENG
IN
10:16
17-12-2024

main.c

Share | Run

```c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
int main() {
    int fd;
    ssize_t bytes;
    char buffer[50];
    fd = open("example.txt", O_CREAT | O_WRONLY, 0644);
    if (fd == -1) {
        perror("Error creating file");
        return 1;
    }
    const char *text = "Hello, UNIX system calls!";
    bytes = write(fd, text, strlen(text));
    if (bytes == -1) {
        perror("Error writing to file");
        close(fd);
        return 1;
    }
    close(fd);
    fd = open("example.txt", O_RDONLY);
    if (fd == -1) {
        perror("Error opening file");
        return 1;
    }
    bytes = read(fd, buffer, sizeof(buffer) - 1);
    if (bytes == -1) {
        perror("Error reading file");
        close(fd);
        return 1;
    }
    buffer[bytes] = '\0'; // Null-terminate the buffer
    printf("Read from file: %s\n", buffer);
    close(fd);
```

Output | Clear

```
A module you have imported isn't available at the moment. It will be available soon.--
```

main.c

Share | Run

```c
    if (fd == -1) {
        perror("Error creating file");
        return 1;
    }
    const char *text = "Hello, UNIX system calls!";
    bytes = write(fd, text, strlen(text));
    if (bytes == -1) {
        perror("Error writing to file");
        close(fd);
        return 1;
    }
    close(fd);
    fd = open("example.txt", O_RDONLY);
    if (fd == -1) {
        perror("Error opening file");
        return 1;
    }
    bytes = read(fd, buffer, sizeof(buffer) - 1);
    if (bytes == -1) {
        perror("Error reading file");
        close(fd);
        return 1;
    }
    buffer[bytes] = '\0'; // Null-terminate the buffer
    printf("Read from file: %s\n", buffer);
    close(fd);
    if (unlink("example.txt") == -1) {
        perror("Error deleting file");
        return 1;
    }

    printf("File operations completed successfully.\n");
    return 0;
}
```

Output | Clear

```
A module you have imported isn't available at the moment. It will be available soon.--
```

main.c

Share | Run

```c
#include<stdio.h>
#include<fcntl.h>
#include<errno.h>
extern int errno; int
main()
{

int fd = open("foo.txt", O_RDONLY | O_CREAT);
printf("fd = %d\n", fd);
if (fd ==-1)
{
printf("Error Number % d\n", errno);
perror("Program");
}
return 0;
}
```

Output | Clear

```
fd = -1
Error Number  13
Program: Permission denied


=== Code Execution Successful ===
```

Sign in | OS_LAB_MANUAL[2].pdf | online c compiler - Search | Online C Compiler - Programiz | 433a2x6c4 - C - OneCompiler | Microsoft Copilot: Your AI com

https://www.programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

main.c

Share | Run

```c
#include <stdio.h>
#include <stdlib.h>
void createFile(const char *filename) {
    FILE *file = fopen(filename, "w");
    if (!file) {
        fprintf(stderr, "Failed to create file: %s\n", filename);
        exit(EXIT_FAILURE);
    }
    fprintf(file, "File created successfully.\n");
    fclose(file);
}
void readFile(const char *filename) {
    char ch;
    FILE *file = fopen(filename, "r");
    if (!file) {
        fprintf(stderr, "Failed to open file: %s\n", filename);
        exit(EXIT_FAILURE);
    }
    while ((ch = fgetc(file)) != EOF) {
        putchar(ch);
    }
    fclose(file);
}
void deleteFile(const char *filename) {
    if (remove(filename) != 0) {
        fprintf(stderr, "Error deleting file: %s\n", filename);
    } else {
        printf("File deleted successfully.\n");
    }
}
int main() {
```

Output | Clear

```
Failed to create file: example.txt


=== Code Exited With Errors ===
```


77°F Mostly cloudy | Search | ENG IN | 10:38 17-12-2024

Sign in | OS_LAB_MANUAL[2].pdf | online c compiler - Search | Online C Compiler - Programiz | 433a2x6c4 - C - OneCompiler | Microsoft Copilot: Your AI com

https://www.programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

main.c

Share | Run

```c
    fprintf(file, "File created successfully.\n");
    fclose(file);
}
void readFile(const char *filename) {
    char ch;
    FILE *file = fopen(filename, "r");
    if (!file) {
        fprintf(stderr, "Failed to open file: %s\n", filename);
        exit(EXIT_FAILURE);
    }
    while ((ch = fgetc(file)) != EOF) {
        putchar(ch);
    }
    fclose(file);
}
void deleteFile(const char *filename) {
    if (remove(filename) != 0) {
        fprintf(stderr, "Error deleting file: %s\n", filename);
    } else {
        printf("File deleted successfully.\n");
    }
}
int main() {
    const char *filename = "example.txt";
    createFile(filename);
    readFile(filename);
    deleteFile(filename);
    return 0;
}
```

Output | Clear

```
Failed to create file: example.txt


=== Code Exited With Errors ===
```


77°F Mostly cloudy | Search | ENG IN | 10:38 17-12-2024

Programiz
C Online Compiler

Programiz PRO >

main.c | Share | Run

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

int main() {
    struct dirent *entry;
    DIR *dp = opendir(".");

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

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

    closedir(dp);
    return 0;
}
```

Output | Clear

```
.
..
.bash_logout
.bashrc
.profile


=== Code Execution Successful ===
```

Sign in | OS_LAB_MANUAL[1].pd | OS_LAB_MANUAL[1].pd | c program compiler - Se | Online C Compiler - Pro | 433fva44h - C - OneCon | FREE AI Code Generator

https://www.programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

main.c

Share | Run

```c
#include <string.h>

void grep(const char *filename, const char *pattern) {
    FILE *file = fopen(filename, "r");
    char line[256];

    if (file == NULL) {
        perror("Error opening file");
        return;
    }

    while (fgets(line, sizeof(line), file)) {
        if (strstr(line, pattern)) {
            printf("%s", line);
        }
    }

    fclose(file);
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: %s <filename> <pattern>\n", argv[0]);
        return 1;
    }

    grep(argv[1], argv[2]);
    return 0;
}
```

Output | Clear

```
Usage: /tmp/eywnSgg9Yi/main.o <filename> <pattern>


=== Code Exited With Errors ===
```


Finance headline 'Wrong-headed':...

Search

ENG IN

08:53 19-12-2024

Programiz
C Online Compiler

Programiz PRO >

main.c

Share | Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define BUFFER_SIZE 5
int buffer[BUFFER_SIZE];
int in = 0, out = 0;
sem_t empty, full, mutex;

// Function for producer thread
void* producer(void* arg) {
    int item;
    while (1) {
        item = rand() % 100;  // Produce a random item
        sem_wait(&empty);     // Decrease empty semaphore
        sem_wait(&mutex);     // Enter critical section

        // Produce an item and add it to the buffer
        buffer[in] = item;
        in = (in + 1) % BUFFER_SIZE;
        printf("Produced: %d\n", item);

        sem_post(&mutex);     // Exit critical section
        sem_post(&full);      // Increase full semaphore

        sleep(1);  // Sleep for 1 second
    }
}
```

Output | Clear

```
Produced: 83
Consumed: 83
Produced: 86
Consumed: 86
Produced: 77
Consumed: 77
Produced: 15
Consumed: 15
Produced: 93
Consumed: 93
Produced: 35
Consumed: 35
Produced: 86
Consumed: 86
Produced: 92
Consumed: 92
Produced: 49
Consumed: 49
Produced: 21
Consumed: 21
Produced: 62
Consumed: 62
Produced: 27
Consumed: 27
Produced: 90
Consumed: 90
Produced: 59
Consumed: 59
Produced: 63
Consumed: 63
```

Programiz
C Online Compiler

Programiz PRO >

main.c

Share | Run

```c
// Function for consumer thread
void* consumer(void* arg) {
    int item;
    while (1) {
        sem_wait(&full);       // Decrease full semaphore
        sem_wait(&mutex);      // Enter critical section

        // Consume an item from the buffer
        item = buffer[out];
        out = (out + 1) % BUFFER_SIZE;
        printf("Consumed: %d\n", item);

        sem_post(&mutex);      // Exit critical section
        sem_post(&empty);      // Increase empty semaphore

        sleep(1);  // Sleep for 1 second
    }
}

int main() {
    // Initialize semaphores
    if (sem_init(&empty, 0, BUFFER_SIZE) != 0) {
        perror("sem_init(empty) failed");
        return -1;
    }
    if (sem_init(&full, 0, 0) != 0) {
        perror("sem_init(full) failed");
        return -1;
    }
    if (sem_init(&mutex, 0, 1) != 0) {
```

Output | Clear

```
Produced: 83
Consumed: 83
Produced: 86
Consumed: 86
Produced: 77
Consumed: 77
Produced: 15
Consumed: 15
Produced: 93
Consumed: 93
Produced: 35
Consumed: 35
Produced: 86
Consumed: 86
Produced: 92
Consumed: 92
Produced: 49
Consumed: 49
Produced: 21
Consumed: 21
Produced: 62
Consumed: 62
Produced: 27
Consumed: 27
Produced: 90
Consumed: 90
Produced: 59
Consumed: 59
Produced: 63
Consumed: 63
```

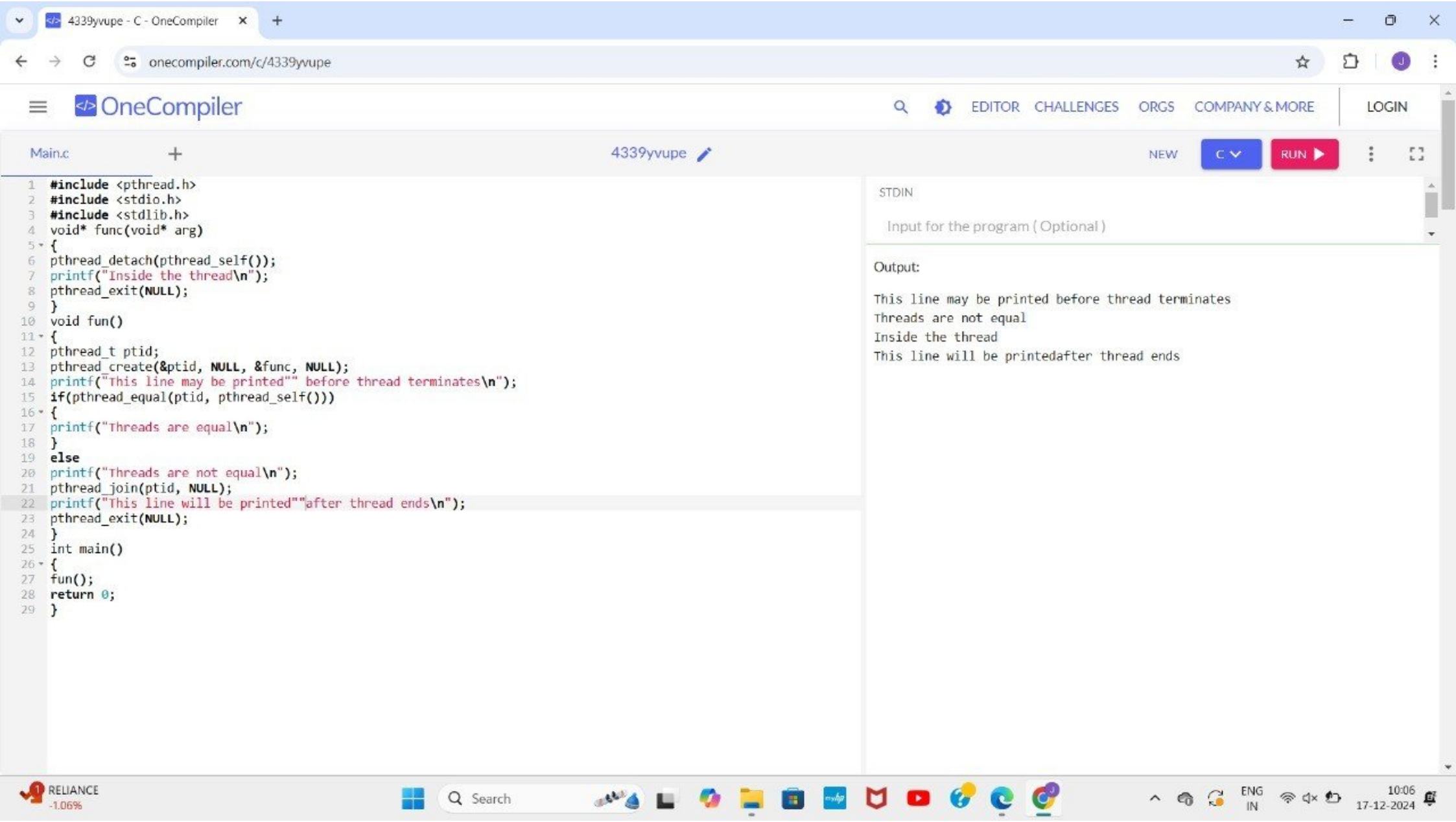

main.c

Share | Run

```c
    }
    if (sem_init(&mutex, 0, 1) != 0) {
        perror("sem_init(mutex) failed");
        return -1;
    }

    pthread_t producer_thread, consumer_thread;

    // Create producer and consumer threads
    if (pthread_create(&producer_thread, NULL, producer, NULL) != 0) {
        perror("Failed to create producer thread");
        return -1;
    }
    if (pthread_create(&consumer_thread, NULL, consumer, NULL) != 0) {
        perror("Failed to create consumer thread");
        return -1;
    }

    // Join threads (this will never happen in this infinite loop scenario)
    pthread_join(producer_thread, NULL);
    pthread_join(consumer_thread, NULL);

    // Destroy semaphores (this is not reached because of infinite loop)
    sem_destroy(&empty);
    sem_destroy(&full);
    sem_destroy(&mutex);

    return 0;
}
```

Output | Clear

```
Produced: 83
Consumed: 83
Produced: 86
Consumed: 86
Produced: 77
Consumed: 77
Produced: 15
Consumed: 15
Produced: 93
Consumed: 93
Produced: 35
Consumed: 35
Produced: 86
Consumed: 86
Produced: 92
Consumed: 92
Produced: 49
Consumed: 49
Produced: 21
Consumed: 21
Produced: 62
Consumed: 62
Produced: 27
Consumed: 27
Produced: 90
Consumed: 90
Produced: 59
Consumed: 59
Produced: 63
Consumed: 63
```

4339yvupe - C - OneCompiler

onecompiler.com/c/4339yvupe

OneCompiler

EDITOR CHALLENGES ORGS COMPANY & MORE LOGIN

Main.c

4339yvupe

NEW C RUN

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
void* func(void* arg)
{
pthread_detach(pthread_self());
printf("Inside the thread\n");
pthread_exit(NULL);
}
void fun()
{
pthread_t ptid;
pthread_create(&ptid, NULL, &func, NULL);
printf("This line may be printed"" before thread terminates\n");
if(pthread_equal(ptid, pthread_self()))
{
printf("Threads are equal\n");
}
else
printf("Threads are not equal\n");
pthread_join(ptid, NULL);
printf("This line will be printed""after thread ends\n");
pthread_exit(NULL);
}
int main()
{
fun();
return 0;
}
```

STDIN

Input for the program (Optional)

Output:

```
This line may be printed before thread terminates
Threads are not equal
Inside the thread
This line will be printedafter thread ends
```

RELIANCE -1.06%

Search

ENG IN

10:06 17-12-2024

4339yvupe - C - OneCompiler

onecompiler.com/c/4339yvupe

OneCompiler

EDITOR CHALLENGES ORGS COMPANY & MORE LOGIN

Main.c

4339yvupe

NEW C RUN

```c
#include <stdio.h>
#define MAX_FRAMES 3
void printFrames(int frames[], int n) {
  for (int i = 0; i < n; i++) {
if (frames[i] == -1) {
printf(" - ");
} else {
printf(" %d ", frames[i]);
}
}
printf("\n");
}
int main() {
int referenceString[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2}; int n =
sizeof(referenceString) / sizeof(referenceString[0]);
int frames[MAX_FRAMES];
int framePointer = 0;
for (int i = 0; i < MAX_FRAMES; i++)
{
frames[i] = -1;
}
printf("Reference String: ");for
(int i = 0; i < n; i++) {
printf("%d ", referenceString[i]);
}printf("\n\n");
printf("Page Replacement Order:\n");
for (int i = 0; i < n; i++) {
int page = referenceString[i];
int pageFound = 0;
for (int j = 0; j < MAX_FRAMES; j++) {
if (frames[j] == page) {
pageFound = 1; break;
}
}
if (!pageFound) {
printf("Page %d -> ", page);
frames[framePointer] = page;
framePointer = (framePointer + 1) % MAX_FRAMES;
printFrames(frames, MAX_FRAMES);
}
}
return 0;
}
```

STDIN

Input for the program (Optional)

Output:

```
Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 2

Page Replacement Order:
Page 7 ->  7  -  -
Page 0 ->  7  0  -
Page 1 ->  7  0  1
Page 2 ->  2  0  1
Page 3 ->  2  3  1
Page 0 ->  2  3  0
Page 4 ->  4  3  0
Page 2 ->  4  2  0
Page 3 ->  4  2  3
Page 0 ->  0  2  3
```

HP Support Assistant

Finance headline
India reported -...

Search

ENG
IN

10:10
17-12-2024

4339yvupe - C - OneCompiler

onecompiler.com/c/4339yvupe

OneCompiler

EDITOR CHALLENGES ORGS COMPANY & MORE LOGIN

Main.c

4339yvupe

NEW C RUN

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_FRAMES 3
void printFrames(int frames[], int n) {
  for (int i = 0; i < n; i++) {
if (frames[i] == -1) {
printf(" - ");
} else {
printf(" %d ", frames[i]);
}
}
printf("\n");
}
int main() {
int frames[MAX_FRAMES];
int usageHistory[MAX_FRAMES];
for (int i = 0; i < MAX_FRAMES; i++) {
frames[i] = -1;
usageHistory[i] = 0;
}
int pageFaults = 0;
int referenceString[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};
int n = sizeof(referenceString) / sizeof(referenceString[0]);
printf("Reference String: ");
for (int i = 0; i < n; i++) {
printf("%d ", referenceString[i]);
}
printf("\n\n");
printf("Page Replacement Order:\n");for
(int i = 0; i < n; i++) {
int page = referenceString[i];int
pageFound = 0;
for (int j = 0; j < MAX_FRAMES; j++) {
if (frames[j] == page) {
pageFound = 1;
for (int k = 0; k < MAX_FRAMES; k++) {
if (k != j) {
usageHistory[k]++;
}
```

STDIN

Input for the program (Optional)

Output:

```
Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 2

Page Replacement Order:
Page 7 -> Load into an empty frame:  7  -  -
Page 0 -> Replace 7 with 0:  0  -  -
Page 1 -> Replace 0 with 1:  1  -  -
Page 2 -> Replace 1 with 2:  2  -  -
Page 0 -> Replace 2 with 0:  0  -  -
Page 3 -> Replace 0 with 3:  3  -  -
Page 0 -> Replace 3 with 0:  0  -  -
Page 4 -> Replace 0 with 4:  4  -  -
Page 2 -> Replace 4 with 2:  2  -  -
Page 3 -> Replace 2 with 3:  3  -  -
Page 0 -> Replace 3 with 0:  0  -  -
Page 3 -> Replace 0 with 3:  3  -  -
Page 2 -> Replace 3 with 2:  2  -  -

Total Page Faults: 13
```

SIMATS
Internet access

SENSEX -0.41%

Search

ENG IN

10:15 17-12-2024

4339yvupe - C - OneCompiler

onecompiler.com/c/4339yvupe

Main.c

4339yvupe

NEW C RUN

```c
int page = referenceString[i];int
pageFound = 0;
for (int j = 0; j < MAX_FRAMES; j++) {
if (frames[j] == page) {
pageFound = 1;
for (int k = 0; k < MAX_FRAMES; k++) {
if (k != j) {
usageHistory[k]++;
}
}
usageHistory[j] = 0; // Reset the usage counter for the used pagebreak;
}
}
if (!pageFound) {
printf("Page %d -> ", page);
// Find the page with the maximum usage counter (least recently
int lruPage = 0;
for (int j = 1; j < MAX_FRAMES; j++) {
if (usageHistory[j] > usageHistory[lruPage])
{
  lruPage= j;
}
}
int replacedPage = frames[lruPage];
frames[lruPage] = page;
usageHistory[lruPage] = 0;
if (replacedPage != -1) {
printf("Replace %d with %d: ", replacedPage, page);
} else {
printf("Load into an empty frame: ");
}
printFrames(frames, MAX_FRAMES);
pageFaults++;
}
}
printf("\nTotal Page Faults: %d\n", pageFaults);
return 0;
}
```

STDIN

Input for the program (Optional)

Output:

```
Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 2

Page Replacement Order:
Page 7 -> Load into an empty frame:  7  -  -
Page 0 -> Replace 7 with 0:  0  -  -
Page 1 -> Replace 0 with 1:  1  -  -
Page 2 -> Replace 1 with 2:  2  -  -
Page 0 -> Replace 2 with 0:  0  -  -
Page 3 -> Replace 0 with 3:  3  -  -
Page 0 -> Replace 3 with 0:  0  -  -
Page 4 -> Replace 0 with 4:  4  -  -
Page 2 -> Replace 4 with 2:  2  -  -
Page 3 -> Replace 2 with 3:  3  -  -
Page 0 -> Replace 3 with 0:  0  -  -
Page 3 -> Replace 0 with 3:  3  -  -
Page 2 -> Replace 3 with 2:  2  -  -

Total Page Faults: 13
```

SIMATS
Internet access

Nifty bank -0.42%

Search

ENG IN

10:15 17-12-2024

4339yvupe - C - OneCompiler

onecompiler.com/c/4339yvupe

OneCompiler

EDITOR CHALLENGES ORGS COMPANY & MORE LOGIN

Main.c

4339yvupe

NEW C RUN

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_FRAMES 3
void printFrames(int frames[], int n)
{
  for (int i = 0; i < n; i++) {
if (frames[i] == -1) {
printf(" - ");
} else {
printf(" %d ", frames[i]);
}
}
printf("\n");
}
int main() {
int frames[MAX_FRAMES];
for (int i = 0; i < MAX_FRAMES; i++) {
  frames[i] = -1;
}
int pageFaults = 0;
int referenceString[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};
int n = sizeof(referenceString) / sizeof(referenceString[0]);
printf("Reference String: ");
for (int i = 0; i < n; i++) {
printf("%d ", referenceString[i]);
}
printf("\n\n");
printf("Page Replacement Order:\n");
for (int i = 0; i < n; i++) {
int page = referenceString[i];
int pageFound = 0;
// Check if the page is already in memory (a page hit)
for (int j = 0; j < MAX_FRAMES; j++) {
if (frames[j] == page) {
pageFound = 1; break;
}
}
if (!pageFound) {
printf("Page %d -> ", page);
```

STDIN

Input for the program (Optional)

Output:

```
Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 2

Page Replacement Order:
Page 7 ->  7  -  -
Page 0 ->  0  -  -
Page 1 ->  0  1  -
Page 2 ->  0  2  -
Page 3 ->  0  2  3
Page 4 ->  4  2  3
Page 0 ->  0  2  3

Total Page Faults: 7
```

Show desktop

25°C Mostly cloudy

Search

ENG IN

10:18 17-12-2024

4339yvupe - C - OneCompiler

onecompiler.com/c/4339yvupe

Main.c

4339yvupe

NEW C RUN

```c
for (int i = 0; i < n; i++) {
printf("%d ", referenceString[i]);
}
printf("\n\n");
printf("Page Replacement Order:\n");
for (int i = 0; i < n; i++) {
int page = referenceString[i];
int pageFound = 0;
// Check if the page is already in memory (a page hit)
for (int j = 0; j < MAX_FRAMES; j++) {
if (frames[j] == page) {
pageFound = 1; break;
}
}
if (!pageFound) {
printf("Page %d -> ", page);
int optimalPage = -1;
int farthestDistance = 0;
for (int j = 0; j < MAX_FRAMES; j++) {
  int futureDistance = 0;
for (int k = i + 1; k < n; k++) {
if (referenceString[k] == frames[j]) {
  break;
}
futureDistance++;
}
if (futureDistance > farthestDistance) {
farthestDistance = futureDistance;
optimalPage = j;
}
}
frames[optimalPage] = page;
printFrames(frames, MAX_FRAMES);
pageFaults++;
}
}
printf("\nTotal Page Faults: %d\n", pageFaults);
return 0;
}
```

STDIN

Input for the program (Optional)

Output:

```
Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 2

Page Replacement Order:
Page 7 ->  7  -  -
Page 0 ->  0  -  -
Page 1 ->  0  1  -
Page 2 ->  0  2  -
Page 3 ->  0  2  3
Page 4 ->  4  2  3
Page 0 ->  0  2  3

Total Page Faults: 7
```

OneDrive - Personal
Paused

C Language online compiler

25°C
Mostly cloudy

Search

ENG
IN

10:18
17-12-2024

4339yvupe - C - OneCompiler | Online C Compiler - Programiz

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

Programiz PRO >

main.c

Share | Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_RECORDS 100
#define RECORD_SIZE 256
typedef struct {
    char data[RECORD_SIZE];
} Record;
typedef struct {
    Record records[MAX_RECORDS];
    int count;
} FileSystem;
void initFileSystem(FileSystem *fs) {
    fs->count = 0;
}
int addRecord(FileSystem *fs, const char *data) {
    if (fs->count >= MAX_RECORDS) {
        printf("Error: File system is full. Cannot add more records.\n");
        return -1;
    }
    strncpy(fs->records[fs->count].data, data, RECORD_SIZE);
    fs->count++;
    return 0;
}
void readRecords(const FileSystem *fs) {
    if (fs->count == 0) {
        printf("No records to read.\n");
        return;
    }
    printf("Reading records:\n");
    for (int i = 0; i < fs->count; i++) {
        printf("Record %d: %s\n", i + 1, fs->records[i].data);
    }
}
int main() {
```

Output | Clear

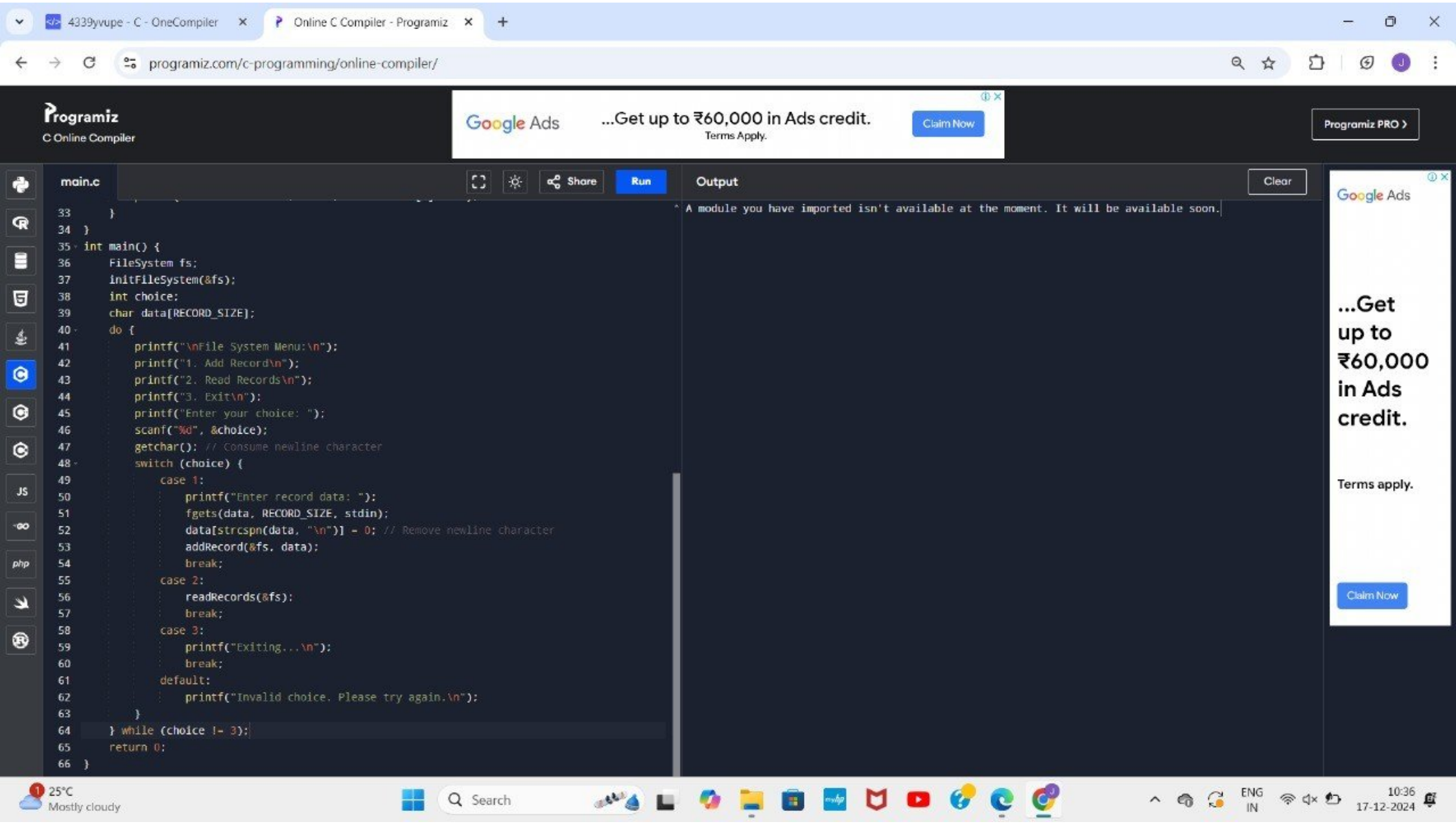
```
A module you have imported isn't available at the moment. It will be available soon.
```


25°C Mostly cloudy

Search

ENG IN

10:36 17-12-2024




```c
    }
}
int main() {
    FileSystem fs;
    initFileSystem(&fs);
    int choice;
    char data[RECORD_SIZE];
    do {
        printf("\nFile System Menu:\n");
        printf("1. Add Record\n");
        printf("2. Read Records\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar(); // Consume newline character
        switch (choice) {
            case 1:
                printf("Enter record data: ");
                fgets(data, RECORD_SIZE, stdin);
                data[strcspn(data, "\n")] = 0; // Remove newline character
                addRecord(&fs, data);
                break;
            case 2:
                readRecords(&fs);
                break;
            case 3:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 3);
    return 0;
}
```

main.c

Share Run

```c
#include <stdio.h>
void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
    wt[0] = 0;
    for (int i = 1; i < n; i++)
        wt[i] = bt[i - 1] + wt[i - 1];
}
void findTurnAroundTime(int processes[], int n, int bt[], int wt[],
    int tat[]) {
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}
void findavgTime(int processes[], int n, int bt[]) {
    int wt[n], tat[n];
    findWaitingTime(processes, n, bt, wt);
    findTurnAroundTime(processes, n, bt, wt, tat);
    printf("Process \tBurst Time\tWaiting Time\tTurn-Around Time\n");
    for (int i = 0; i < n; i++)
        printf("%d \t%d\t\t%d\t\t%d\n", processes[i], bt[i], wt[i],
            tat[i]);
}
int main() {
    int processes[] = {1, 2, 3}; // Process IDs
    int n = sizeof(processes) / sizeof(processes[0]);
    int burst_time[] = {10, 5, 8}; // Burst times
    findavgTime(processes, n, burst_time);
    return 0;
```

Output

Clear

```
Process     Burst Time  Waiting Time    Turn-Around Time
1   10      0       10
2   5       10      15
3   8       15      23


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct Block {
    int data;
    struct Block* next;
} Block;
typedef struct File {
    Block* head;
    Block* tail;
} File;
File* createFile() {
    File* file = (File*)malloc(sizeof(File));
    file->head = file->tail = NULL;
    return file;
}
void addBlock(File* file, int data) {
    Block* newBlock = (Block*)malloc(sizeof(Block));
    newBlock->data = data;
    newBlock->next = NULL;
    if (!file->head) {
        file->head = file->tail = newBlock;
    } else {
        file->tail->next = newBlock;
        file->tail = newBlock;
    }
}
```

Output Clear

```
1 -> 2 -> 3 -> NULL


=== Code Execution Successful ===
```

main.c

Share Run

```
    if (!file->head) {
        file->head = file->tail = newBlock;
    } else {
        file->tail->next = newBlock;
        file->tail = newBlock;
    }
}

void displayFile(File* file) {
    Block* current = file->head;
    while (current) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    File* myFile = createFile();
    addBlock(myFile, 1);
    addBlock(myFile, 2);
    addBlock(myFile, 3);
    displayFile(myFile);
    return 0;
}

```

Output Clear

```
1 -> 2 -> 3 -> NULL


=== Code Execution Successful ===
```

main.c

Share Run

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

#define BLOCKS 5
typedef struct {
    int blockPointers[BLOCKS];
} IndexBlock;
void allocateFile(IndexBlock *index) {
    for (int i = 0; i < BLOCKS; i++) {
        index->blockPointers[i] = i + 1;
    }
}
void displayAllocation(IndexBlock *index) {
    printf("File Allocation:\n");
    for (int i = 0; i < BLOCKS; i++) {
        printf("Block %d -> Pointer %d\n", i, index->blockPointers[i]);
    }
}
int main() {
    IndexBlock index;
    allocateFile(&index);
    displayAllocation(&index);
    return 0;
}

```

Output Clear

```
File Allocation:
Block 0 -> Pointer 1
Block 1 -> Pointer 2
Block 2 -> Pointer 3
Block 3 -> Pointer 4
Block 4 -> Pointer 5


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>
void SCAN(int arr[], int size, int head, int direction) {
    int seek_sequence[size + 1], index = 0, distance, cur_track;
    int total_head_movement = 0, start = 0, end = size - 1;
    for (int i = 0; i < size - 1; i++)
        for (int j = 0; j < size - i - 1; j++)
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
    if (direction == 1) {
        for (int i = start; i < size; i++) {
            if (arr[i] >= head) {
                seek_sequence[index++] = head;
                distance = arr[i] - head;
                total_head_movement += distance;
                head = arr[i];
            }
        }
        seek_sequence[index++] = head;
        total_head_movement += (arr[end] - head);
        head = arr[end];
        for (int i = end; i >= start; i--) {
            if (arr[i] < head) {
                seek_sequence[index++] = head;
```

Output Clear

```
Total head movement: 270
Seek Sequence: 100 100 160 180 200 200 180 160 100 90 50

=== Code Execution Successful ===
```

main.c

Share Run

```
            distance = head - arr[i];
            total_head_movement += distance;
            head = arr[i];
        }
    }
} else {
    for (int i = end; i >= start; i--) {
        if (arr[i] <= head) {
            seek_sequence[index++] = head;
            distance = head - arr[i];
            total_head_movement += distance;
            head = arr[i];
        }
    }
    seek_sequence[index++] = head;
    total_head_movement += (head - arr[start]);
    head = arr[start];
    for (int i = start; i < size; i++) {
        if (arr[i] > head) {
            seek_sequence[index++] = head;
            distance = arr[i] - head;
            total_head_movement += distance;
            head = arr[i];
        }
    }
}
```

Output

Clear

```
Total head movement: 270
Seek Sequence: 100 100 160 180 200 200 180 160 100 90 50

=== Code Execution Successful ===
```

main.c

Share Run

```
        seek_sequence[index++] = head;
        total_head_movement += (head - arr[start]);
        head = arr[start];
        for (int i = start; i < size; i++) {
            if (arr[i] > head) {
                seek_sequence[index++] = head;
                distance = arr[i] - head;
                total_head_movement += distance;
                head = arr[i];
            }
        }
    }
    printf("Total head movement: %d\n", total_head_movement);
    printf("Seek Sequence: ");
    for (int i = 0; i < index; i++)
        printf("%d ", seek_sequence[i]);
}
int main() {
    int arr[] = { 100, 180, 50, 30, 200, 90, 160 };
    int size = sizeof(arr) / sizeof(arr[0]);
    int head = 100; // Initial head position
    int direction = 1; // 1 for right, 0 for left
    SCAN(arr, size, head, direction);
    return 0;
```

Output

```
Total head movement: 270
Seek Sequence: 100 100 160 180 200 200 180 160 100 90 5

=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>

void cscan(int arr[], int size, int head, int total_cylinders) {
    int seek_sequence[size + 1];
    int index = 0;

    // Move to the right
    for (int i = head; i < total_cylinders; i++) {
        for (int j = 0; j < size; j++) {
            if (arr[j] == i) {
                seek_sequence[index++] = arr[j];
            }
        }
    }

    // Jump to the start
    seek_sequence[index++] = 0;

    // Move to the left
    for (int i = 1; i < head; i++) {
        for (int j = 0; j < size; j++) {
            if (arr[j] == i) {
                seek_sequence[index++] = arr[j];
            }
        }
    }
```

Output

Clear

```
Seek Sequence: 60 79 92 114 176 0 11 34 41


=== Code Execution Successful ===
```

main.c

Share Run

```c
    // Move to the left
    for (int i = 1; i < head; i++) {
        for (int j = 0; j < size; j++) {
            if (arr[j] == i) {
                seek_sequence[index++] = arr[j];
            }
        }
    }

    printf("Seek Sequence: ");
    for (int i = 0; i < index; i++) {
        printf("%d ", seek_sequence[i]);
    }
    printf("\n");
}

int main() {
    int requests[] = {176, 79, 34, 60, 92, 11, 41, 114};
    int size = sizeof(requests) / sizeof(requests[0]);
    int head = 50;
    int total_cylinders = 200;

    cscan(requests, size, head, total_cylinders);
    return 0;
}
```

Output

Clear

```
Seek Sequence: 60 79 92 114 176 0 11 34 41


=== Code Execution Successful ===
```

main.c

Share Run

```c
#include <stdio.h>
#include <sys/stat.h>

void print_permissions(mode_t mode) {
    printf("Permissions: ");
    printf((mode & S_IRUSR) ? "r" : "-");
    printf((mode & S_IWUSR) ? "w" : "-");
    printf((mode & S_IXUSR) ? "x" : "-");
    printf((mode & S_IRGRP) ? "r" : "-");
    printf((mode & S_IWGRP) ? "w" : "-");
    printf((mode & S_IXGRP) ? "x" : "-");
    printf((mode & S_IROTH) ? "r" : "-");
    printf((mode & S_IWOTH) ? "w" : "-");
    printf((mode & S_IXOTH) ? "x" : "-");
    printf("\n");
}

int main() {
    struct stat fileStat;
    stat("example.txt", &fileStat);
    print_permissions(fileStat.st_mode);
    return 0;
}
```

Output

Clear

```
Permissions: ---------

=== Code Execution Successful ===
```