Exp:23

Code:

#include <stdio.h>

#define MAX\_MEMORY 1000

// Structure to represent a memory block

struct MemoryBlock {

int size;

int allocated; // 0 for unallocated, 1 for allocated

};

// Function to initialize the memory blocks

void initializeMemory(struct MemoryBlock memory[], int size) {

for (int i = 0; i < size; i++) {

memory[i].size = 0;

memory[i].allocated = 0;

}

}

// Function to display the current state of memory

void displayMemory(struct MemoryBlock memory[], int size) {

printf("Memory State:\n");

for (int i = 0; i < size; i++) {

printf("Block %d: Size=%d, Allocated=%s\n", i, memory[i].size,

memory[i].allocated ? "Yes" : "No");

}

printf("\n");

}

// Function to allocate memory using First Fit algorithm

int allocateMemory(struct MemoryBlock memory[], int size, int requestSize) {

for (int i = 0; i < size; i++) {

if (!memory[i].allocated && memory[i].size >= requestSize) {

memory[i].allocated = 1;

return i; // Return the index of the allocated block

}

}

return -1; // No suitable block found

}

int main() {

struct MemoryBlock memory[MAX\_MEMORY];

int memorySize, requestSize, blockIndex;

printf("Enter the size of memory: ");

scanf("%d", &memorySize);

initializeMemory(memory, memorySize);

while (1) {

displayMemory(memory, memorySize);

printf("Enter the size of memory request (or enter 0 to exit): ");

scanf("%d", &requestSize);

if (requestSize == 0) {

printf("Exiting the program.\n");

break;

}

blockIndex = allocateMemory(memory, memorySize, requestSize);

if (blockIndex != -1) {

printf("Memory allocated successfully in block %d.\n", blockIndex);

} else {

printf("No suitable block found for allocation.\n");

}

}

return 0;

}

Output:

