OS-Lab

Lab#14

Name: Anas-Altaf

Roll.No: 22F-3639

C++ Codes: Memory Management

```
#include <iostream>
#include <vector>
#include <climits>
#include <cstdlib>
#include <ctime>

const int TOTAL_MEMORY_SIZE = 100;

struct MemoryBlock {
    int start_address;
    int size;
    bool is_allocated;
};

class MemoryAllocator {
private:
    MemoryBlock memory_array[TOTAL_MEMORY_SIZE];

public:
    MemoryAllocator() {
        srand((unsigned int)(time(NULL)));
        for (int i = 0; i < TOTAL_MEMORY_SIZE; ++i) {
            memory_array[i].start_address = i;
            memory_array[i].size = 1;
            memory_array[i].is_allocated = rand() % 2; // Randomly
allocate or free
        }
```

```
    }

    bool firstFitAllocate(int requested_size) {
        int contiguousFreeBlock = 0;
        for (int i = 0; i < TOTAL_MEMORY_SIZE; ++i) {
            if (!memory_array[i].is_allocated) {
                contiguousFreeBlock++;
                if (contiguousFreeBlock == requested_size) {
                    for (int j = i - requested_size + 1; j <= i; ++j) {
                        memory_array[j].is_allocated = true;
                    }
                    std::cout << "First-Fit: Allocated " << requested_size
<< " bytes | at address " << (i - requested_size + 1) << std::endl;
                    return true;
                }
            } else {
                contiguousFreeBlock = 0;
            }
        }
        std::cout << "First-Fit: Unable to allocate " << requested_size <<
" bytes" << std::endl;
        return false;
    }

    bool bestFitAllocate(int requested_size) {
        int bestFitStart = -1, smallestRequiredBlock = INT_MAX;
        for (int i = 0; i < TOTAL_MEMORY_SIZE; ++i) {
            if (!memory_array[i].is_allocated) {
                int contiguousFreeBlock = 0, start_index = i;
                while (i < TOTAL_MEMORY_SIZE &&
!memory_array[i].is_allocated) {
                    contiguousFreeBlock++;
                    i++;
                }
                if (contiguousFreeBlock >= requested_size &&
contiguousFreeBlock < smallestRequiredBlock) {
                    bestFitStart = start_index;
                    smallestRequiredBlock = contiguousFreeBlock;
                }
            }
```

```
        }
        if (bestFitStart != -1) {
            for (int j = bestFitStart; j < bestFitStart + requested_size;
++j) {
                memory_array[j].is_allocated = true;
            }
            std::cout << "Best-Fit: Allocated " << requested_size << "
bytes | at address " << bestFitStart << std::endl;
            return true;
        }
        std::cout << "Best-Fit: Unable to find size of " << requested_size
<< " bytes" << std::endl;
        return false;
    }

    bool worstFitAllocate(int requested_size) {
        int startOfWorst = -1, findingFreeBlockNum = -1;
        for (int i = 0; i < TOTAL_MEMORY_SIZE; ++i) {
            if (!memory_array[i].is_allocated) {
                int contiguousFreeBlock = 0, start_index = i;
                while (i < TOTAL_MEMORY_SIZE &&
!memory_array[i].is_allocated) {
                    contiguousFreeBlock++;
                    i++;
                }
                if (contiguousFreeBlock >= requested_size &&
contiguousFreeBlock > findingFreeBlockNum) {
                    startOfWorst = start_index;
                    findingFreeBlockNum = contiguousFreeBlock;
                }
            }
        }
        if (startOfWorst != -1) {
            for (int j = startOfWorst; j < startOfWorst + requested_size;
++j) {
                memory_array[j].is_allocated = true;
            }
            std::cout << "Worst-Fit: Allocated " << requested_size << "
bytes | at address " << startOfWorst << std::endl;
            return true;
```

```
        }
        std::cout << "Worst-Fit: No Space Found for " << requested_size <<
" bytes" << std::endl;
        return false;
    }

    void printMemoryState() {
        std::cout << "Memory State:\n";
        for (int i = 0; i < TOTAL_MEMORY_SIZE; ++i) {
            std::cout << "Address: " << i << ", Allocated: " <<
memory_array[i].is_allocated << "\n";
        }
    }
};

int main() {
    MemoryAllocator allocator;
    int choice, size;
    while (true) {
        std::cout << "\nMemory Allocator Menu:\n";
        std::cout << "1. First-Fit Allocation\n";
        std::cout << "2. Best-Fit Allocation\n";
        std::cout << "3. Worst-Fit Allocation\n";
        std::cout << "4. Print Memory State\n";
        std::cout << "5. Exit\n";
        std::cout << "Enter your choice: ";
        std::cin >> choice;

        if (choice == 5) {
            std::cout << "Exiting the program.\n";
            break;
        }

        if (choice >= 1 && choice <= 3) {
            std::cout << "Enter the size of memory to allocate: ";
            std::cin >> size;

            switch (choice) {
                case 1:
                    allocator.firstFitAllocate(size);
```

```
                    break;
                case 2:
                    allocator.bestFitAllocate(size);
                    break;
                case 3:
                    allocator.worstFitAllocate(size);
                    break;
                default:
                    std::cout << "Invalid choice. Please try again.\n";
            }
        } else if (choice == 4) {
            allocator.printMemoryState();
        } else {
            std::cout << "Invalid choice. Please try again.\n";
        }
    }

    return 0;
}
```

Output :

root@ns3-virtual-machine:/home/ns3/lab_14_22f3639# ./a.out

Memory Allocator Menu:
1. First-Fit Allocation
2. Best-Fit Allocation
3. Worst-Fit Allocation
4. Print Memory State
5. Exit
Enter your choice: 1
Enter the size of memory to allocate: 4
First-Fit: Allocated 4 bytes | at address 20

Memory Allocator Menu:
1. First-Fit Allocation
2. Best-Fit Allocation
3. Worst-Fit Allocation
4. Print Memory State
5. Exit
Enter your choice: 2

Enter the size of memory to allocate: 3
Best-Fit: Allocated 3 bytes | at address 45

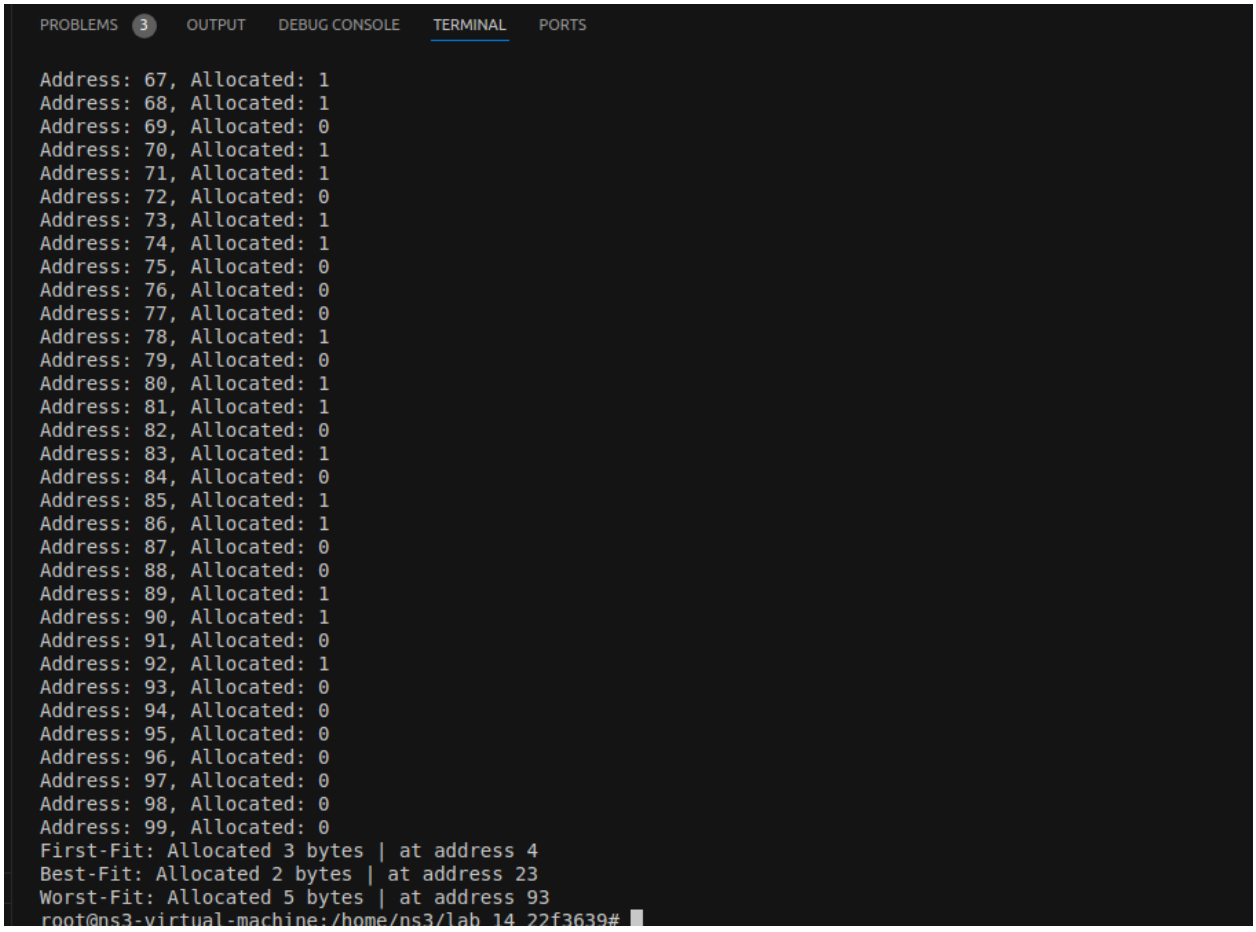
Memory Allocator Menu:
1. First-Fit Allocation
2. Best-Fit Allocation
3. Worst-Fit Allocation
4. Print Memory State
5. Exit
Enter your choice: 3
Enter the size of memory to allocate: 4
Worst-Fit: No Space Found for 4 bytes

Memory Allocator Menu:
1. First-Fit Allocation
2. Best-Fit Allocation
3. Worst-Fit Allocation
4. Print Memory State
5. Exit
Enter your choice: 4
Memory State:
Address: 0, Allocated: 0
Address: 1, Allocated: 1
Address: 2, Allocated: 0
Address: 3, Allocated: 1
Address: 4, Allocated: 1
Address: 5, Allocated: 0
Address: 6, Allocated: 1
Address: 7, Allocated: 0
Address: 8, Allocated: 0
Address: 9, Allocated: 1
Address: 10, Allocated: 0
Address: 11, Allocated: 1
Address: 12, Allocated: 1
Address: 13, Allocated: 1
Address: 14, Allocated: 1
Address: 15, Allocated: 1
Address: 16, Allocated: 1
Address: 17, Allocated: 0
Address: 18, Allocated: 1
Address: 19, Allocated: 1
Address: 20, Allocated: 1
Address: 21, Allocated: 1
Address: 22, Allocated: 1

Address: 23, Allocated: 1
Address: 24, Allocated: 1
Address: 25, Allocated: 1
Address: 26, Allocated: 0
Address: 27, Allocated: 1
Address: 28, Allocated: 1
Address: 29, Allocated: 0
Address: 30, Allocated: 1
Address: 31, Allocated: 1
Address: 32, Allocated: 0
Address: 33, Allocated: 1
Address: 34, Allocated: 1
Address: 35, Allocated: 1
Address: 36, Allocated: 1
Address: 37, Allocated: 0
Address: 38, Allocated: 0
Address: 39, Allocated: 1
Address: 40, Allocated: 0
Address: 41, Allocated: 0
Address: 42, Allocated: 1
Address: 43, Allocated: 1
Address: 44, Allocated: 1
Address: 45, Allocated: 1
Address: 46, Allocated: 1
Address: 47, Allocated: 1
Address: 48, Allocated: 1
Address: 49, Allocated: 1
Address: 50, Allocated: 1
Address: 51, Allocated: 1
Address: 52, Allocated: 1
Address: 53, Allocated: 0
Address: 54, Allocated: 1
Address: 55, Allocated: 1
Address: 56, Allocated: 1
Address: 57, Allocated: 1
Address: 58, Allocated: 0
Address: 59, Allocated: 0
Address: 60, Allocated: 0
Address: 61, Allocated: 1
Address: 62, Allocated: 0
Address: 63, Allocated: 0
Address: 64, Allocated: 0
Address: 65, Allocated: 1
Address: 66, Allocated: 1

Address: 67, Allocated: 1
Address: 68, Allocated: 1
Address: 69, Allocated: 1
Address: 70, Allocated: 1
Address: 71, Allocated: 1
Address: 72, Allocated: 0
Address: 73, Allocated: 0
Address: 74, Allocated: 0
Address: 75, Allocated: 1
Address: 76, Allocated: 0
Address: 77, Allocated: 1
Address: 78, Allocated: 0
Address: 79, Allocated: 1
Address: 80, Allocated: 0
Address: 81, Allocated: 1
Address: 82, Allocated: 1
Address: 83, Allocated: 0
Address: 84, Allocated: 1
Address: 85, Allocated: 0
Address: 86, Allocated: 1
Address: 87, Allocated: 1
Address: 88, Allocated: 0
Address: 89, Allocated: 1
Address: 90, Allocated: 1
Address: 91, Allocated: 0
Address: 92, Allocated: 1
Address: 93, Allocated: 1
Address: 94, Allocated: 0
Address: 95, Allocated: 1
Address: 96, Allocated: 0
Address: 97, Allocated: 1
Address: 98, Allocated: 1
Address: 99, Allocated: 0

Memory Allocator Menu:
1. First-Fit Allocation
2. Best-Fit Allocation
3. Worst-Fit Allocation
4. Print Memory State
5. Exit
Enter your choice:


PROBLEMS 3
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Address: 67, Allocated: 1
Address: 68, Allocated: 1
Address: 69, Allocated: 0
Address: 70, Allocated: 1
Address: 71, Allocated: 1
Address: 72, Allocated: 0
Address: 73, Allocated: 1
Address: 74, Allocated: 1
Address: 75, Allocated: 0
Address: 76, Allocated: 0
Address: 77, Allocated: 0
Address: 78, Allocated: 1
Address: 79, Allocated: 0
Address: 80, Allocated: 1
Address: 81, Allocated: 1
Address: 82, Allocated: 0
Address: 83, Allocated: 1
Address: 84, Allocated: 0
Address: 85, Allocated: 1
Address: 86, Allocated: 1
Address: 87, Allocated: 0
Address: 88, Allocated: 0
Address: 89, Allocated: 1
Address: 90, Allocated: 1
Address: 91, Allocated: 0
Address: 92, Allocated: 1
Address: 93, Allocated: 0
Address: 94, Allocated: 0
Address: 95, Allocated: 0
Address: 96, Allocated: 0
Address: 97, Allocated: 0
Address: 98, Allocated: 0
Address: 99, Allocated: 0
First-Fit: Allocated 3 bytes | at address 4
Best-Fit: Allocated 2 bytes | at address 23
Worst-Fit: Allocated 5 bytes | at address 93
root@ns3-virtual-machine:/home/ns3/lab_14_22f3639#