OPERATING SYSTEM

ASSIGNMENT-6

Prabhnoor Singh
3NC3
102115059

Q1. To simulate the following contiguous memory allocation Techniques
a) Worst fit
b) Best fit
c) First fit.

```cpp
#include <iostream>
#include <vector>

using namespace std;

struct MemoryBlock {
    int id;
    int size;
    bool allocated;
};

vector<MemoryBlock> memory;

void initializeMemory(int totalMemorySize) {
    memory.clear();
    MemoryBlock block;
    block.id = 0;
    block.size = totalMemorySize;
    block.allocated = false;
    memory.push_back(block);
}

bool allocateFirstFit(int processSize) {
    for (int i = 0; i < memory.size(); i++) {
        if (!memory[i].allocated && memory[i].size >= processSize) {
            if (memory[i].size > processSize) {
                MemoryBlock newBlock;
                newBlock.id = memory.size();
```

```
                newBlock.size = memory[i].size - processSize;
                newBlock.allocated = false;
                memory.insert(memory.begin() + i + 1, newBlock);
            }
            memory[i].size = processSize;
            memory[i].allocated = true;
            cout << "Allocated at position " << i << endl;
            return true;
        }
    }
    return false;
}

bool allocateBestFit(int processSize) {
    int bestFit = -1;
    for (int i = 0; i < memory.size(); i++) {
        if (!memory[i].allocated && memory[i].size >= processSize) {
            if (bestFit == -1 || memory[i].size < memory[bestFit].size) {
                bestFit = i;
            }
        }
    }
    if (bestFit != -1) {
        if (memory[bestFit].size > processSize) {
            MemoryBlock newBlock;
            newBlock.id = memory.size();
            newBlock.size = memory[bestFit].size - processSize;
            newBlock.allocated = false;
            memory.insert(memory.begin() + bestFit + 1, newBlock);
        }
        memory[bestFit].size = processSize;
        memory[bestFit].allocated = true;
        cout << "Allocated at position " << bestFit << endl;
        return true;
    }
    return false;
}

bool allocateWorstFit(int processSize) {
    int worstFit = -1;
    for (int i = 0; i < memory.size(); i++) {
        if (!memory[i].allocated && memory[i].size >= processSize) {
            if (worstFit == -1 || memory[i].size > memory[worstFit].size) {
                worstFit = i;
            }
        }
    }
    if (worstFit != -1) {
        if (memory[worstFit].size > processSize) {
            MemoryBlock newBlock;
            newBlock.id = memory.size();
            newBlock.size = memory[worstFit].size - processSize;
            newBlock.allocated = false;
            memory.insert(memory.begin() + worstFit + 1, newBlock);
```

```
        }
        memory[worstFit].size = processSize;
        memory[worstFit].allocated = true;
        cout << "Allocated at position " << worstFit << endl;
        return true;
    }
    return false;
}

void displayMemory() {
    for (int i = 0; i < memory.size(); i++) {
        cout << "Block " << memory[i].id << ": ";
        if (memory[i].allocated) {
            cout << "Allocated (Size: " << memory[i].size << ")" << endl;
        } else {
            cout << "Free (Size: " << memory[i].size << ")" << endl;
        }
    }
}

int main() {
    int totalMemorySize;
    cout << "Enter total memory size: ";
    cin >> totalMemorySize;
    initializeMemory(totalMemorySize);

    while (true) {
        int choice, processSize;
        cout << "1. Allocate (First Fit)" << endl;
        cout << "2. Allocate (Best Fit)" << endl;
        cout << "3. Allocate (Worst Fit)" << endl;
        cout << "4. Display Memory" << endl;
        cout << "5. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter process size: ";
                cin >> processSize;
                if (!allocateFirstFit(processSize)) {
                    cout << "Allocation failed." << endl;
                }
                break;
            case 2:
                cout << "Enter process size: ";
                cin >> processSize;
                if (!allocateBestFit(processSize)) {
                    cout << "Allocation failed." << endl;
                }
                break;
            case 3:
                cout << "Enter process size: ";
                cin >> processSize;
```

```
                if (!allocateWorstFit(processSize)) {
                    cout << "Allocation failed." << endl;
                }
                break;
            case 4:
                displayMemory();
                break;
            case 5:
                return 0;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    }

    return 0;
}
```

Output:

```
vboxuser@Ubuntu:~$ editor code.cpp
vboxuser@Ubuntu:~$ g++ code.cpp -o code
vboxuser@Ubuntu:~$ ./code
Enter total memory size: 100
1. Allocate (First Fit)
2. Allocate (Best Fit)
3. Allocate (Worst Fit)
4. Display Memory
5. Exit
Enter your choice: 1
Enter process size: 30
Allocated at position 0
1. Allocate (First Fit)
2. Allocate (Best Fit)
3. Allocate (Worst Fit)
4. Display Memory
5. Exit
Enter your choice: 2
Enter process size: 20
Allocated at position 1
1. Allocate (First Fit)
2. Allocate (Best Fit)
3. Allocate (Worst Fit)
4. Display Memory
5. Exit
Enter your choice: 3
Enter process size: 25
Allocated at position 2
1. Allocate (First Fit)
2. Allocate (Best Fit)
3. Allocate (Worst Fit)
4. Display Memory
5. Exit
Enter your choice: 4
Block 0: Allocated (Size: 30)
Block 1: Allocated (Size: 20)
Block 2: Allocated (Size: 25)
Block 3: Free (Size: 25)
1. Allocate (First Fit)
2. Allocate (Best Fit)
3. Allocate (Worst Fit)
4. Display Memory
5. Exit
Enter your choice: 5
```