

**Lab No: 9 Date: 2082/**

**Title: Write a program to display the process allocation for user input free blocks and incoming process using First fit allocation.**

The First Fit algorithm allocates the first available memory block that is large enough to satisfy a process’s request. It is simple and fast because it stops searching as soon as a suitable block is found. However, it may lead to external fragmentation over time.

Algorithm:

Step 1: Start with a list of memory blocks and a list of processes with their sizes.

Step 2: Select the first process from the list of processes.

Step 3: Scan the memory blocks from the beginning to the end.

Step 4: Find the first memory block that is large enough to accommodate the selected process.

Step 5: Allocate the process to that memory block.

Step 6: Reduce the available size of the memory block by the size of the allocated process.

Step 7: If no suitable block is found, mark the process as unallocated.

Step 8: Repeat Steps 2–7 for all remaining processes.

Step 9: Stop when all processes are allocated or attempted for allocation.

**Language**: C++

**IDE**: VS Code

**Code:**

**#include <iostream>**

**#include <vector>**

**using namespace std;**

**void worstFit(vector<int>& blockSize, const vector<int>& processSize) {**

**int n = processSize.size(); // number of processes**

**int m = blockSize.size();   // number of blocks**

**vector<int> allocation(n, -1);**

**// Keep original block sizes for later display**

**vector<int> blockSizeBefore = blockSize;**

**for (int i = 0; i < n; i++) {**

**int worstIdx = -1;**

**for (int j = 0; j < m; j++) {**

**if (blockSize[j] >= processSize[i]) {**

**if (worstIdx == -1 || blockSize[j] > blockSize[worstIdx]) {**

**worstIdx = j;**

**}**

**}**

**}**

**if (worstIdx != -1) {**

**allocation[i] = worstIdx;**

**blockSize[worstIdx] -= processSize[i]; // Allocate**

**}**

**}**

**// Final output without Remaining Space**

**cout << "\nAllocation Result:\n";**

**cout << "Process No.\tProcess Size\tBlock No.\n";**

**cout << "-------------------------------------------\n";**

**for (int i = 0; i < n; i++) {**

**cout << i + 1 << "\t\t" << processSize[i] << "\t\t";**

**if (allocation[i] != -1) {**

**cout << allocation[i] + 1;**

**} else {**

**cout << "Not Allocated";**

**}**

**cout << endl;**

**}**

**// Show final state of all blocks**

**cout << "\nFinal State of Blocks:\n";**

**cout << "Block No.\tOriginal Size\tRemaining Space\n";**

**cout << "--------------------------------------------\n";**

**for (int j = 0; j < m; j++) {**

**cout << j + 1 << "\t\t" << blockSizeBefore[j] << "\t\t" << blockSize[j] << endl;**

**}**

**}**

**int main() {**

**int m, n;**

**cout << "Enter number of free memory blocks: ";**

**cin >> m;**

**vector<int> blockSize(m);**

**cout << "Enter the size of each free memory block:\n";**

**for (int i = 0; i < m; i++) {**

**cout << "Block " << i + 1 << ": ";**

**cin >> blockSize[i];**

**}**

**cout << "\nEnter number of processes: ";**

**cin >> n;**

**vector<int> processSize(n);**

**cout << "Enter the size of each process:\n";**

**for (int i = 0; i < n; i++) {**

**cout << "Process " << i + 1 << ": ";**

**cin >> processSize[i];**

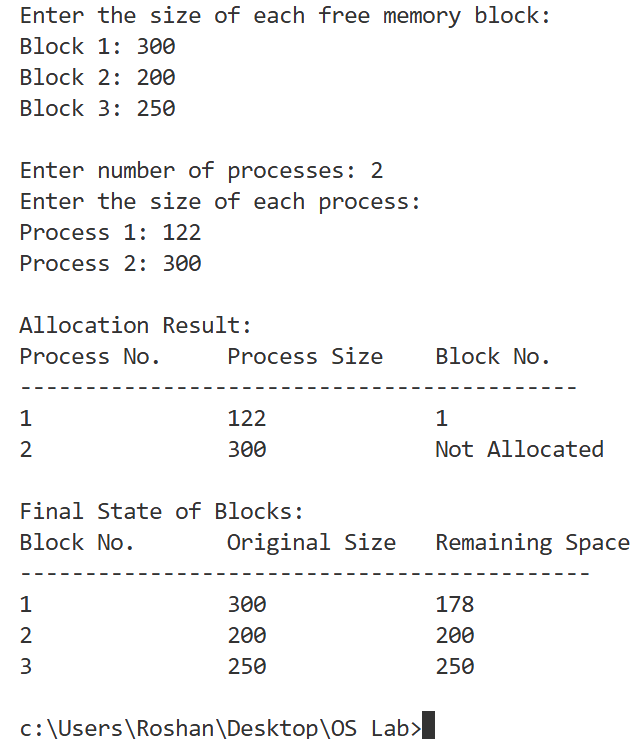
**}**

**worstFit(blockSize, processSize);**

**return 0;**

**}**

**Output:**

****

**Conclusion:**

First Fit allocates each process to the first suitable memory block. It is simple and fast but can cause external fragmentation. It works well for quick allocation in small to medium systems.