12. Write a C program to implement the best-fit algorithm and allocate the memory block to each process.

Test Case:

Memory partitions: 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order),

Show the outcome for the test case with the best-fit algorithms to place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)

#include <stdio.h>

#include <stdlib.h>

#define NUM\_PARTITIONS 6

#define NUM\_PROCESSES 5

int main() {

int partitions[NUM\_PARTITIONS] = {300, 600, 350, 200, 750, 125}; // Memory partitions

int processes[NUM\_PROCESSES] = {115, 500, 358, 200, 375}; // Processes to allocate

int allocated[NUM\_PROCESSES] = {0}; // Array to track which processes have been allocated

int i, j, best\_fit\_index;

// Iterate through each process to allocate

for (i = 0; i < NUM\_PROCESSES; i++) {

best\_fit\_index = -1; // Initialize the best fit index to -1

// Find the partition with the smallest size that can fit the current process

for (j = 0; j < NUM\_PARTITIONS; j++) {

if (partitions[j] >= processes[i]) {

if (best\_fit\_index == -1 || partitions[j] < partitions[best\_fit\_index]) {

best\_fit\_index = j;

}

}

}

// If a suitable partition was found, allocate the process to that partition

if (best\_fit\_index != -1) {

partitions[best\_fit\_index] -= processes[i];

allocated[i] = 1;

}

}

// Print the allocation results

printf("Allocation results:\n");

for (i = 0; i < NUM\_PROCESSES; i++) {

if (allocated[i]) {

printf("Process %d allocated\n", processes[i]);

} else {

printf("Process %d not allocated\n", processes[i]);

}

}

return 0;

}

