**7. Implement memory allocation method for fixed partition using first fit algorithm.**

#include <stdio.h>

#include <stdbool.h>

#define MAX\_PARTITIONS 10

#define MAX\_PROCESSES 10111

struct Partition {

int id;

int size;

bool allocated;

};

struct Process {

int id;

int size;

int partition\_id; // ID of the allocated partition

};

struct Partition partitions[MAX\_PARTITIONS];

struct Process processes[MAX\_PROCESSES];

int num\_partitions = 0;

int num\_processes = 0;

void initializePartitions() {

partitions[0].id = 1;

partitions[0].size = 50;

partitions[0].allocated = false;

partitions[1].id = 2;

partitions[1].size = 100;

partitions[1].allocated = false;

partitions[2].id = 3;

partitions[2].size = 500;

partitions[2].allocated = false;

num\_partitions = 3; // Update number of partitions

}

void initializeProcesses() {

// Example processes initialization (you can modify this)

processes[0].id = 1;

processes[0].size = 80;

processes[1].id = 2;

processes[1].size = 150;

num\_processes = 2; // Update number of processes

}

void displayPartitions() {

printf("Partitions:\n");

printf("ID\tSize\tAllocated\n");

for (int i = 0; i < num\_partitions; i++) {

printf("%d\t%d\t%s\n", partitions[i].id, partitions[i].size, partitions[i].allocated ? "Yes" : "No");

}

printf("\n");

}

void displayProcesses() {

printf("Processes:\n");

printf("ID\tSize\tPartition ID\n");

for (int i = 0; i < num\_processes; i++) {

printf("%d\t%d\t%d\n", processes[i].id, processes[i].size,

processes[i].partition\_id);

}

printf("\n");

}

void firstFit() {

for (int i = 0; i < num\_processes; i++) {

for (int j = 0; j < num\_partitions; j++) {

if (!partitions[j].allocated && partitions[j].size >= processes[i].size) {

// Allocate the partition to the process

partitions[j].allocated = true;

processes[i].partition\_id = partitions[j].id;

printf("Process %d allocated to Partition %d\n", processes[i].id, partitions[j].id);

break; // Break out of inner loop once allocation is done

}

}

}

}

int main() {

initializePartitions();

initializeProcesses();

printf("Initial State:\n");

displayPartitions();

displayProcesses();

firstFit();

printf("After Allocation:\n");

displayPartitions();

displayProcesses();

return 0;

}

**OUTPUT:**

Initial State:

Partitions:

ID Size Allocated

1 50 No

2 100 No

3 500 No

Processes:

ID Size Partition ID

1 80 0

2 150 0

Process 1 allocated to Partition 2

Process 2 allocated to Partition 3

After Allocation:

Partitions:

ID Size Allocated

1 50 No

2 100 Yes

3 500 Yes

Processes:

ID Size Partition ID

1 80 2

2 150 3