ASSIGNMENT 6

PROGRAM:

#include <iostream>

using namespace std;

struct Block {

int size;

bool allocated;

};

struct Process {

int size;

int allocatedBlock;

};

void firstFit(Block blocks[], int numBlocks, Process processes[], int numProcesses) {

for (int i = 0; i < numProcesses; i++) {

processes[i].allocatedBlock = -1;

for (int j = 0; j < numBlocks; j++) {

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

processes[i].allocatedBlock = j;

blocks[j].allocated = true;

break;

}

}

}

}

void bestFit(Block blocks[], int numBlocks, Process processes[], int numProcesses) {

for (int i = 0; i < numProcesses; i++) {

int bestIdx = -1;

int minSize = 1e9;

for (int j = 0; j < numBlocks; j++) {

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

bestIdx = j;

minSize = blocks[j].size;

}

}

if (bestIdx != -1) {

processes[i].allocatedBlock = bestIdx;

blocks[bestIdx].allocated = true;

}

}

}

void worstFit(Block blocks[], int numBlocks, Process processes[], int numProcesses) {

for (int i = 0; i < numProcesses; i++) {

int worstIdx = -1;

int maxSize = -1;

for (int j = 0; j < numBlocks; j++) {

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

worstIdx = j;

maxSize = blocks[j].size;

}

}

if (worstIdx != -1) {

processes[i].allocatedBlock = worstIdx;

blocks[worstIdx].allocated = true;

}

}

}

void nextFit(Block blocks[], int numBlocks, Process processes[], int numProcesses) {

static int lastAllocIdx = 0;

for (int i = 0; i < numProcesses; i++) {

processes[i].allocatedBlock = -1;

int count = 0, j = lastAllocIdx;

while (count < numBlocks) {

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

processes[i].allocatedBlock = j;

blocks[j].allocated = true;

lastAllocIdx = (j + 1) % numBlocks;

break;

}

j = (j + 1) % numBlocks;

count++;

}

}

}

void printResults(Process processes[], int numProcesses) {

cout << "\nProcess Allocation Results:\n";

for (int i = 0; i < numProcesses; i++) {

cout << "Process " << i + 1 << " (Size " << processes[i].size << ") -> ";

if (processes[i].allocatedBlock != -1)

cout << "Block " << processes[i].allocatedBlock + 1 << "\n";

else

cout << "Not Allocated\n";

}

}

int main() {

int numBlocks, numProcesses;

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

cin >> numBlocks;

Block blocks[numBlocks];

cout << "Enter block sizes:\n";

for (int i = 0; i < numBlocks; i++) {

cin >> blocks[i].size;

blocks[i].allocated = false;

}

cout << "Enter number of processes: ";

cin >> numProcesses;

Process processes[numProcesses];

cout << "Enter process sizes:\n";

for (int i = 0; i < numProcesses; i++) {

cin >> processes[i].size;

}

int choice;

do {

for (int i = 0; i < numBlocks; i++) blocks[i].allocated = false;

for (int i = 0; i < numProcesses; i++) processes[i].allocatedBlock = -1;

cout << "\nMemory Allocation Techniques:\n";

cout << "1. First Fit\n2. Best Fit\n3. Worst Fit\n4. Next Fit\n5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

firstFit(blocks, numBlocks, processes, numProcesses);

printResults(processes, numProcesses);

break;

case 2:

bestFit(blocks, numBlocks, processes, numProcesses);

printResults(processes, numProcesses);

break;

case 3:

worstFit(blocks, numBlocks, processes, numProcesses);

printResults(processes, numProcesses);

break;

case 4:

nextFit(blocks, numBlocks, processes, numProcesses);

printResults(processes, numProcesses);

break;

case 5:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice! Try again.\n";

}

} while (choice != 5);

return 0;

}

OUTPUT:

PS C:\Users\User\Desktop\OSL> g++ osl6.cpp

PS C:\Users\User\Desktop\OSL> .\a

Enter number of memory blocks: 4

Enter block sizes:

200

20

700

50

Enter number of processes: 4

Enter process sizes:

20

200

500

50

Memory Allocation Techniques:

1. First Fit

2. Best Fit

3. Worst Fit

4. Next Fit

5. Exit

Enter your choice: 1

Process Allocation Results:

Process 1 (Size 20) -> Block 1

Process 2 (Size 200) -> Block 3

Process 3 (Size 500) -> Not Allocated

Process 4 (Size 50) -> Block 4

Memory Allocation Techniques:

1. First Fit

2. Best Fit

3. Worst Fit

4. Next Fit

5. Exit

Enter your choice: 2

Process Allocation Results:

Process 1 (Size 20) -> Block 2

Process 2 (Size 200) -> Block 1

Process 3 (Size 500) -> Block 3

Process 4 (Size 50) -> Block 4

Memory Allocation Techniques:

1. First Fit

2. Best Fit

3. Worst Fit

4. Next Fit

5. Exit

Enter your choice: 3

Process Allocation Results:

Process 1 (Size 20) -> Block 3

Process 2 (Size 200) -> Block 1

Process 3 (Size 500) -> Not Allocated

Process 4 (Size 50) -> Block 4

Memory Allocation Techniques:

1. First Fit

2. Best Fit

3. Worst Fit

4. Next Fit

5. Exit

Enter your choice: 4

Process Allocation Results:

Process 1 (Size 20) -> Block 1

Process 2 (Size 200) -> Block 3

Process 3 (Size 500) -> Not Allocated

Process 4 (Size 50) -> Block 4

Memory Allocation Techniques:

1. First Fit

2. Best Fit

3. Worst Fit

4. Next Fit

5. Exit

Enter your choice: 5

Exiting...

PS C:\Users\User\Desktop\OSL>