**Name : Ruturaj Sandip Sutar Roll No : 59 Div : B**

**Batch : 2 PRN:-12310720**

**Implementation of memory placement strategies**

1. **First Fit**

**Code :-**

*#include* <iostream>

using namespace std;

void firstFit(int *blockSize*[], int *m*, int *processSize*[], int *n*) {

    int allocation[*n*];

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

        allocation[i] = -1;

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

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

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

                allocation[i] = j;

*blockSize*[j] -= *processSize*[i];

*break*;

            }

        }

    }

    cout << "\nProcess No.\tProcess Size\tBlock No.\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 << endl;

*else*

            cout << "Not Allocated" << endl;

    }

}

int main() {

    int blockSize[] = {100, 500, 200, 300, 600};

    int processSize[] = {212, 417, 112, 426};

    int m = sizeof(blockSize) / sizeof(blockSize[0]);

    int n = sizeof(processSize) / sizeof(processSize[0]);

    firstFit(blockSize, m, processSize, n);

*return* 0;

}

**Output:-**

Process No. Process Size Block No.

1 212 2

2 417 5

3 112 2

4 426 Not Allocated

1. **Next Fit**

**Code :-**

*#include* <iostream>

using namespace std;

void nextFit(int *blockSize*[], int *m*, int *processSize*[], int *n*) {

    int allocation[*n*];

    int j = 0; *// To keep track of the last allocated block*

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

        allocation[i] = -1;

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

        int start = j; *// Store the starting position for each process*

        bool allocated = false;

*while* (true) {

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

                allocation[i] = j;

*blockSize*[j] -= *processSize*[i];

                allocated = true;

*break*;

            }

            j = (j + 1) % *m*;

*// If we return to the start, that means we've tried all blocks*

*if* (j == start) {

*break*;

            }

        }

*// Move to the next block for the next process*

*if* (allocated) {

            j = (j + 1) % *m*;

        }

    }

    cout << "\nProcess No.\tProcess Size\tBlock No.\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 << endl;

*else*

            cout << "Not Allocated" << endl;

    }

}

int main() {

    int blockSize[] = {100, 500, 200, 300, 600};

    int processSize[] = {212, 417, 112, 426};

    int m = sizeof(blockSize) / sizeof(blockSize[0]);

    int n = sizeof(processSize) / sizeof(processSize[0]);

    nextFit(blockSize, m, processSize, n);

*return* 0;

}

**Output:-**

Process No. Process Size Block No.

1 212 2

2 417 5

3 112 2

4 426 Not Allocated

1. **Best Fit**

**Code:-**

*#include* <iostream>

*#include* <climits>

using namespace std;

void bestFit(int *blockSize*[], int *m*, int *processSize*[], int *n*) {

    int allocation[*n*];

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

        allocation[i] = -1;

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

        int bestIdx = -1;

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

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

*if*(bestIdx == -1 || *blockSize*[j] < *blockSize*[bestIdx]) {

                    bestIdx = j;

                }

            }

        }

*if*(bestIdx != -1) {

            allocation[i] = bestIdx;

*blockSize*[bestIdx] -= *processSize*[i];

        }

    }

    cout << "\nProcess No.\tProcess Size\tBlock No.\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 << endl;

*else*

            cout << "Not Allocated" << endl;

    }

}

int main() {

    int blockSize[] = {100, 500, 200, 300, 600};

    int processSize[] = {212, 417, 112, 426};

    int m = sizeof(blockSize) / sizeof(blockSize[0]);

    int n = sizeof(processSize) / sizeof(processSize[0]);

    bestFit(blockSize, m, processSize, n);

*return* 0;

}

**Output:-**

Process No. Process Size Block No.

1 212 4

2 417 2

3 112 3

4 426 5

1. **Worst Fit**

**Code:-**

*#include* <iostream>

using namespace std;

void worstFit(int *blockSize*[], int *m*, int *processSize*[], int *n*) {

    int allocation[*n*];

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

        allocation[i] = -1;

*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];

        }

    }

    cout << "\nProcess No.\tProcess Size\tBlock No.\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 << endl;

*else*

            cout << "Not Allocated" << endl;

    }

}

int main() {

    int blockSize[] = {100, 500, 200, 300, 600};

    int processSize[] = {212, 417, 112, 426};

    int m = sizeof(blockSize) / sizeof(blockSize[0]);

    int n = sizeof(processSize) / sizeof(processSize[0]);

    worstFit(blockSize, m, processSize, n);

*return* 0;

}

**Output:-**

Process No. Process Size Block No.

1 212 5

2 417 2

3 112 5

4 426 Not Allocated