EXPERIMENT NO. 10

SIMULATION OF INTERNAL AND EXTERNAL FRAGMENTATION

Objective

To write a program to simulate internal and external fragmentation according to:
a) First Fit
b) Worst Fit
c) Best Fit allocation strategies.

Theory

Memory allocation strategies determine how processes are assigned to blocks of memory. The most common strategies include First Fit, Best Fit, and Worst Fit.

Strategies

- First Fit: Allocates the first block that is large enough.
- Best Fit: Allocates the smallest block that is sufficient.
- Worst Fit: Allocates the largest available block.

Internal Fragmentation: Wasted space within an allocated block.
External Fragmentation: Total wasted space due to small unallocated gaps.

Program Code (C Language)

```
#include <stdio.h>

void firstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n], i, j;
    for(i = 0; i < n; i++) allocation[i] = -1;

    for(i = 0; i < n; i++) {
        for(j = 0; j < m; j++) {
            if(blockSize[j] >= processSize[i]) {
                allocation[i] = j;
                blockSize[j] -= processSize[i];
                break;
            }
        }
    }

    printf("First Fit Allocation:\n");
    for(i = 0; i < n; i++) {
        printf("Process %d -> Block %d\n", i+1, allocation[i]+1);
    }
}

void bestFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n], i, j, bestIdx;
    for(i = 0; i < n; i++) allocation[i] = -1;

    for(i = 0; i < n; i++) {
        bestIdx = -1;
        for(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];
        }
    }

    printf("Best Fit Allocation:\n");
    for(i = 0; i < n; i++) {
        printf("Process %d -> Block %d\n", i+1, allocation[i]+1);
    }
}

void worstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n], i, j, worstIdx;
    for(i = 0; i < n; i++) allocation[i] = -1;
    for(i = 0; i < n; i++) {
        worstIdx = -1;
        for(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];
        }
    }

    printf("Worst Fit Allocation:\n");
    for(i = 0; i < n; i++) {
        printf("Process %d -> Block %d\n", i+1, allocation[i]+1);
    }
}

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);
    bestFit(blockSize, m, processSize, n);
    worstFit(blockSize, m, processSize, n);

    return 0;
}
```

Sample Output

```
First Fit Allocation:

Process 1 -> Block 2

Process 2 -> Block 5

Process 3 -> Block 2

Process 4 -> Block 0

Best Fit Allocation:

Process 1 -> Block 4

Process 2 -> Block 0

Process 3 -> Block 2

Process 4 -> Block 0

Worst Fit Allocation:

Process 1 -> Block 0

Process 2 -> Block 0

Process 3 -> Block 3

Process 4 -> Block 0
```

Conclusion

This program demonstrates the simulation of internal and external fragmentation using First Fit, Best Fit, and Worst Fit algorithms. Each strategy affects memory utilization differently.