Name :- Prem Rajendra Rajput PRN :- 23062701242053

Experiment No. 8
Implementation of Memory Allocation Algorithms

Program :-

```
#include <stdio.h>
void firstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    int blockUsed[m];
    for (int i = 0; i < n; i++)
        allocation[i] = -1;
    for (int i = 0; i < m; i++)
        blockUsed[i] = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (!blockUsed[j] && blockSize[j] >= processSize[i]) {
                allocation[i] = j;
                blockUsed[j] = 1;
                break;
            }
        }
    }
    printf("\nFirst Fit Allocation (no splitting):\n");
    printf("Process\tSize\tBlock\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}
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];
        }
    }
    printf("\nBest Fit Allocation (with splitting):\n");
    printf("Process\tSize\tBlock\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}
void worstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    int blockUsed[m];
    for (int i = 0; i < n; i++)
        allocation[i] = -1;
    for (int i = 0; i < m; i++)
        blockUsed[i] = 0;
    for (int i = 0; i < n; i++) {
        int worstIdx = -1;
        for (int j = 0; j < m; j++) {
            if (!blockUsed[j] && blockSize[j] >= processSize[i]) {
                if (worstIdx == -1 || blockSize[j] > blockSize[worstIdx])
                    worstIdx = j;
            }
        }
        if (worstIdx != -1) {
            allocation[i] = worstIdx;
            blockUsed[worstIdx] = 1;
        }
        }
        printf("\nWorst Fit Allocation (no splitting):\n");
        printf("Process\tSize\tBlock\n");
        for (int i = 0; i < n; i++) {
            printf("%d\t%d\t", i + 1, processSize[i]);
            if (allocation[i] != -1)
                printf("%d\n", allocation[i] + 1);
            else
                printf("Not Allocated\n");
        }
    }
    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]);
        int blocks1[m], blocks2[m], blocks3[m];
        for (int i = 0; i < m; i++) {
            blocks1[i] = blockSize[i];
            blocks2[i] = blockSize[i];
            blocks3[i] = blockSize[i];
        }
        firstFit(blocks1, m, processSize, n);
        bestFit(blocks2, m, processSize, n);
        worstFit(blocks3, m, processSize, n);
        return 0;
}
```

Output :-

```
First Fit Allocation (no splitting):
Process      Size   Block
1 212   2
2 417   5
3 112   3
4 426   Not Allocated
```

```
Best Fit Allocation (with splitting):
Process         Size   Block
1 212    4
2 417    2
3 112    3
4 426    5

Worst Fit Allocation (no splitting):
Process         Size   Block
1 212    5
2 417    2
3 112    4
4 426    Not Allocated
```