**EXPERIMENT -13**

13. Construct a C program for implementation the various memory allocation strategies.

#include <stdio.h>

#define MAX\_BLOCKS 10

#define MAX\_PROCESSES 10

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

int allocation[MAX\_PROCESSES];

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;

}

}

}

printf("\nFirst Fit Allocation:\n");

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

printf("Process %d (Size %d) --> ", i + 1, processSize[i]);

if (allocation[i] != -1)

printf("Block %d\n", allocation[i] + 1);

else

printf("Not Allocated\n");

}

}

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

int allocation[MAX\_PROCESSES];

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:\n");

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

printf("Process %d (Size %d) --> ", i + 1, processSize[i]);

if (allocation[i] != -1)

printf("Block %d\n", allocation[i] + 1);

else

printf("Not Allocated\n");

}

}

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

int allocation[MAX\_PROCESSES];

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];

}

}

printf("\nWorst Fit Allocation:\n");

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

printf("Process %d (Size %d) --> ", i + 1, processSize[i]);

if (allocation[i] != -1)

printf("Block %d\n", allocation[i] + 1);

else

printf("Not Allocated\n");

}

}

int main() {

int blockSize[MAX\_BLOCKS], processSize[MAX\_PROCESSES];

int m, n;

printf("Enter number of memory blocks: ");

scanf("%d", &m);

printf("Enter sizes of %d memory blocks:\n", m);

for (int i = 0; i < m; i++)

scanf("%d", &blockSize[i]);

printf("Enter number of processes: ");

scanf("%d", &n);

printf("Enter sizes of %d processes:\n", n);

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

scanf("%d", &processSize[i]);

int blockCopy1[MAX\_BLOCKS], blockCopy2[MAX\_BLOCKS], blockCopy3[MAX\_BLOCKS];

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

blockCopy1[i] = blockCopy2[i] = blockCopy3[i] = blockSize[i];

}

firstFit(blockCopy1, m, processSize, n);

bestFit(blockCopy2, m, processSize, n);

worstFit(blockCopy3, m, processSize, n);

return 0;

}

SAMPLE INPUT:

Enter number of memory blocks: 5

Enter sizes of 5 memory blocks:

100 500 200 300 600

Enter number of processes: 4

Enter sizes of 4 processes:

212 417 112 426

SAMPLE OUTPUT:

First Fit Allocation:

Process 1 (Size 212) --> Block 2

Process 2 (Size 417) --> Block 5

Process 3 (Size 112) --> Block 2

Process 4 (Size 426) --> Not Allocated

Best Fit Allocation:

Process 1 (Size 212) --> Block 3

Process 2 (Size 417) --> Block 2

Process 3 (Size 112) --> Block 1

Process 4 (Size 426) --> Block 5

Worst Fit Allocation:

Process 1 (Size 212) --> Block 5

Process 2 (Size 417) --> Block 2

Process 3 (Size 112) --> Block 5

Process 4 (Size 426) --> Not Allocated