**EXPERIMENT – 22**

22. Construct a C program to implement best fit algorithm of memory management.

#include <stdio.h>

#define MAX 10

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

int allocation[MAX];

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("\nProcess No.\tProcess Size\tBlock No.\n");

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

printf(" %d\t\t %d\t\t", i + 1, processSize[i]);

if (allocation[i] != -1)

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

else

printf("Not Allocated\n");

}

}

int main() {

int blockSize[MAX], processSize[MAX];

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]);

}

bestFit(blockSize, 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:

Process No. Process Size Block No.

1 212 4

2 417 2

3 112 3

4 426 5