**Name: Atharva Salitri Division: CSAI B**

**Roll No.: 37 PRN: 12310120**

**Subject: OS Lab Assignment 7**

**Title: Implementation of Placement Strategies**

**1) First Fit**

**Code:**

#include <stdio.h>

int main() {

static int block\_arr[10], file\_arr[10];

int fragments[10], blocks[10], files[10];

int m, n, number\_of\_blocks, number\_of\_files, temp;

printf("\nEnter the Total Number of Blocks:\t");

scanf("%d", &number\_of\_blocks);

printf("Enter the Total Number of Files:\t");

scanf("%d", &number\_of\_files);

printf("\nEnter the Size of the Blocks:\n");

for (m = 0; m < number\_of\_blocks; m++) {

printf("Block No.[%d]:\t", m + 1);

scanf("%d", &blocks[m]);

block\_arr[m] = 0;

}

printf("\nEnter the Size of the Files:\n");

for (m = 0; m < number\_of\_files; m++) {

printf("File No.[%d]:\t", m + 1);

scanf("%d", &files[m]);

file\_arr[m] = -1;

}

for (m = 0; m < number\_of\_files; m++) {

temp = -1;

for (n = 0; n < number\_of\_blocks; n++) {

if (block\_arr[n] != 1) {

temp = blocks[n] - files[m];

if (temp >= 0) {

file\_arr[m] = n;

fragments[m] = temp;

block\_arr[n] = 1;

break;

}

}

}

if (file\_arr[m] == -1) {

fragments[m] = -1; // No allocation possible, set fragment to -1

}

}

printf("\nFile Number\tBlock Number\tFile Size\tBlock Size\tFragment");

for (m = 0; m < number\_of\_files; m++) {

printf("\n%d\t\t", m + 1);

if (file\_arr[m] != -1) {

printf("%d\t\t%d\t\t%d\t\t%d",

file\_arr[m] + 1,

files[m],

blocks[file\_arr[m]],

fragments[m]);

} else {

printf("-\t\t%d\t\t-\t\t-", files[m]); // Indicate not allocated

}

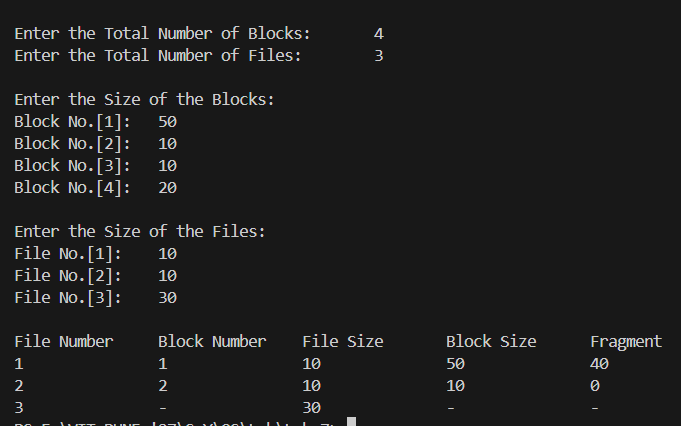
}

printf("\n");

return 0;

}

**Output:**



**2) Next Fit**

**Code:**

#include <stdio.h>

int main() {

static int block\_arr[10], file\_arr[10];

int fragments[10], blocks[10], Process[10];

int m, n, number\_of\_blocks, number\_of\_process, last\_allocated = 0;

printf("\nEnter the Total Number of Blocks:\t");

scanf("%d", &number\_of\_blocks);

printf("Enter the Total Number of Process:\t");

scanf("%d", &number\_of\_process);

printf("\nEnter the Size of the Blocks:\n");

for (m = 0; m < number\_of\_blocks; m++) {

printf("Block No.[%d]:\t", m + 1);

scanf("%d", &blocks[m]);

block\_arr[m] = 0;

}

printf("\nEnter the Size of the Process:\n");

for (m = 0; m < number\_of\_process; m++) {

printf("File No.[%d]:\t", m + 1);

scanf("%d", &Process[m]);

file\_arr[m] = -1;

}

for (m = 0; m < number\_of\_process; m++) {

int allocated = 0; // Flag to check if file is allocated

for (n = last\_allocated; n < number\_of\_blocks; n++) {

if (block\_arr[n] != 1) {

int temp = blocks[n] - Process[m];

if (temp >= 0) {

file\_arr[m] = n;

fragments[m] = temp;

block\_arr[n] = 1;

last\_allocated = n; // Update last allocated position

allocated = 1; // Mark as allocated

break;

}

}

}

// If not allocated in the first pass, check from the beginning

if (!allocated) {

for (n = 0; n < last\_allocated; n++) {

if (block\_arr[n] != 1) {

int temp = blocks[n] - Process[m];

if (temp >= 0) {

file\_arr[m] = n;

fragments[m] = temp;

block\_arr[n] = 1;

last\_allocated = n; // Update last allocated position

allocated = 1; // Mark as allocated

break;

}

}

}

}

if (file\_arr[m] == -1) {

fragments[m] = -1; // No allocation possible, set fragment to -1

}

}

printf("\nFile Number\tBlock Number\tFile Size\tBlock Size\tFragment");

for (m = 0; m < number\_of\_process; m++) {

printf("\n%d\t\t", m + 1);

if (file\_arr[m] != -1) {

printf("%d\t\t%d\t\t%d\t\t%d",

file\_arr[m] + 1,

Process[m],

blocks[file\_arr[m]],

fragments[m]);

} else {

printf("-\t\t%d\t\t-\t\t-", Process[m]); // Indicate not allocated

}

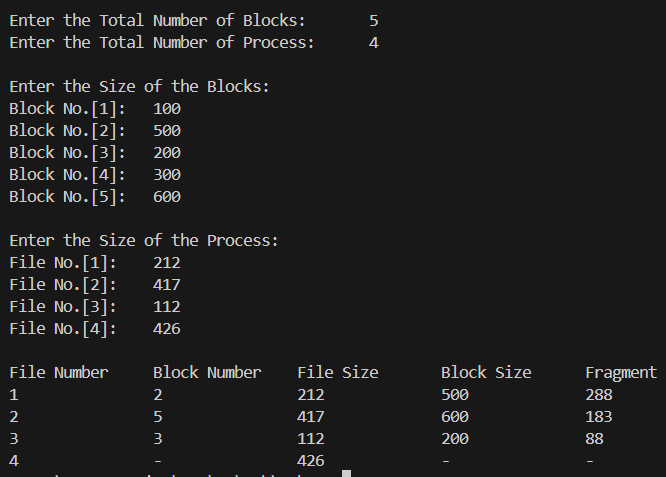
}

printf("\n");

return 0;

}

**Output:**



**3) Best Fit**

**Code:**

#include <stdio.h>

int main() {

static int block\_arr[10], file\_arr[10];

int fragments[10], blocks[10], Process[10];

int m, n, number\_of\_blocks, number\_of\_process;

printf("\nEnter the Total Number of Blocks:\t");

scanf("%d", &number\_of\_blocks);

printf("Enter the Total Number of Processes:\t");

scanf("%d", &number\_of\_process);

printf("\nEnter the Size of the Blocks:\n");

for (m = 0; m < number\_of\_blocks; m++) {

printf("Block No.[%d]:\t", m + 1);

scanf("%d", &blocks[m]);

block\_arr[m] = 0;

}

printf("\nEnter the Size of the Processes:\n");

for (m = 0; m < number\_of\_process; m++) {

printf("Process No.[%d]:\t", m + 1);

scanf("%d", &Process[m]);

file\_arr[m] = -1;

}

for (m = 0; m < number\_of\_process; m++) {

int best\_fit\_index = -1;

int min\_fragment = 9999;

for (n = 0; n < number\_of\_blocks; n++) {

if (block\_arr[n] != 1) { // Check if block is free

int temp\_fragment = blocks[n] - Process[m];

if (temp\_fragment >= 0 && temp\_fragment < min\_fragment) {

best\_fit\_index = n;

min\_fragment = temp\_fragment;

}

}

}

if (best\_fit\_index != -1) {

file\_arr[m] = best\_fit\_index;

fragments[m] = min\_fragment;

block\_arr[best\_fit\_index] = 1;

} else {

fragments[m] = -1; // No allocation possible, set fragment to -1

}

}

printf("\nProcess Number\tBlock Number\tProcess Size\tBlock Size\tFragment");

for (m = 0; m < number\_of\_process; m++) {

printf("\n%d\t\t", m + 1);

if (file\_arr[m] != -1) {

printf("%d\t\t%d\t\t%d\t\t%d",

file\_arr[m] + 1,

Process[m],

blocks[file\_arr[m]],

fragments[m]);

} else {

printf("-\t\t%d\t\t-\t\t-", Process[m]); // Indicate not allocated

}

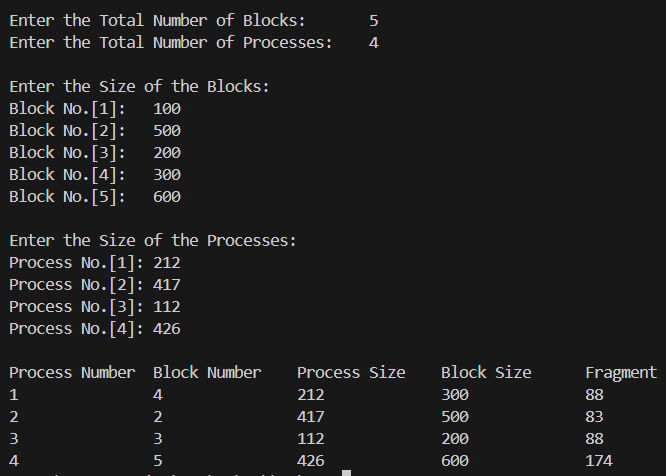
}

printf("\n");

return 0;

}

**Output:**

****

**4) Worst Fit**

**Code:**

#include <stdio.h>

int main() {

static int block\_arr[10], file\_arr[10];

int fragments[10], blocks[10], Process[10];

int m, n, number\_of\_blocks, number\_of\_process;

printf("\nEnter the Total Number of Blocks:\t");

scanf("%d", &number\_of\_blocks);

printf("Enter the Total Number of Processes:\t");

scanf("%d", &number\_of\_process);

printf("\nEnter the Size of the Blocks:\n");

for (m = 0; m < number\_of\_blocks; m++) {

printf("Block No.[%d]:\t", m + 1);

scanf("%d", &blocks[m]);

block\_arr[m] = 0;

}

printf("\nEnter the Size of the Processes:\n");

for (m = 0; m < number\_of\_process; m++) {

printf("Process No.[%d]:\t", m + 1);

scanf("%d", &Process[m]);

file\_arr[m] = -1;

}

for (m = 0; m < number\_of\_process; m++) {

int worst\_fit\_index = -1; // To track the index of the worst fit block

int max\_fragment = -1;

for (n = 0; n < number\_of\_blocks; n++) {

if (block\_arr[n] != 1) {

int temp\_fragment = blocks[n] - Process[m];

if (temp\_fragment >= 0 && temp\_fragment > max\_fragment) {

worst\_fit\_index = n;

max\_fragment = temp\_fragment;

}

}

}

if (worst\_fit\_index != -1) {

file\_arr[m] = worst\_fit\_index;

fragments[m] = max\_fragment;

block\_arr[worst\_fit\_index] = 1;

} else {

fragments[m] = -1; // No allocation possible, set fragment to -1

}

}

printf("\nProcess Number\tBlock Number\tProcess Size\tBlock Size\tFragment");

for (m = 0; m < number\_of\_process; m++) {

printf("\n%d\t\t", m + 1);

if (file\_arr[m] != -1) {

printf("%d\t\t%d\t\t%d\t\t%d",

file\_arr[m] + 1,

Process[m],

blocks[file\_arr[m]],

fragments[m]);

} else {

printf("-\t\t%d\t\t-\t\t-", Process[m]); // Indicate not allocated

}

}

printf("\n");

return 0;

}

**Output:**

