NAME: G. GANESH

REG NO: 192373008

EXERICSE-13

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

Aim:

To construct a C program that implements various memory allocation strategies, including **First Fit**, **Best Fit**, and **Worst Fit**.

Algorithm:

1. **Input Process and Block Sizes:**
   - **Accept the number of memory blocks and their sizes.**
   - **Accept the number of processes and their memory requirements.**
2. **Memory Allocation Strategies:**
   - **FirstFit:**
     Allocate the first available memory block that is large enough.
   - **BestFit:**
     Allocate the smallest block that is large enough.
   - **WorstFit:**
     Allocate the largest block available.
3. **Allocation:**
   - **Iterate through the memory blocks to allocate memory to processes based on the selected strategy.**
   - **Mark the block as allocated if used.**
4. **Display Allocation Results:**
   - **Display the memory allocation for each process.**

Code:

```
#include <stdio.h>

#include <limits.h>

void first_fit(int block_size[], int block_count, int process_size[], int process_count) {

    int allocation[process_count];

    for (int i = 0; i < process_count; i++) {
```

```
        allocation[i] = -1; // Initialize allocation
        for (int j = 0; j < block_count; j++) {
            if (block_size[j] >= process_size[i]) {
                allocation[i] = j; // Allocate block
                block_size[j] -= process_size[i];
                break;
            }
        }
    }
    printf("First Fit Allocation:\n");
    for (int i = 0; i < process_count; i++) {
        if (allocation[i] != -1)
            printf("Process %d -> Block %d\n", i + 1, allocation[i] + 1);
        else
            printf("Process %d -> Not Allocated\n", i + 1);
    }
}
void best_fit(int block_size[], int block_count, int process_size[], int process_count) {
    int allocation[process_count];
    for (int i = 0; i < process_count; i++) {
        allocation[i] = -1;
        int best_index = -1;
        for (int j = 0; j < block_count; j++) {
            if (block_size[j] >= process_size[i]) {
                if (best_index == -1 || block_size[j] < block_size[best_index]) {
                    best_index = j;
                }
            }
        }
        if (best_index != -1) {
```

```
            allocation[i] = best_index;
            block_size[best_index] -= process_size[i];
        }
    }
    printf("Best Fit Allocation:\n");
    for (int i = 0; i < process_count; i++) {
        if (allocation[i] != -1)
            printf("Process %d -> Block %d\n", i + 1, allocation[i] + 1);
        else
            printf("Process %d -> Not Allocated\n", i + 1);
    }
}
void worst_fit(int block_size[], int block_count, int process_size[], int process_count) {
    int allocation[process_count];
    for (int i = 0; i < process_count; i++) {
        allocation[i] = -1;
        int worst_index = -1;
        for (int j = 0; j < block_count; j++) {
            if (block_size[j] >= process_size[i]) {
                if (worst_index == -1 || block_size[j] > block_size[worst_index]) {
                    worst_index = j;
                }
            }
        }
        if (worst_index != -1) {
            allocation[i] = worst_index;
            block_size[worst_index] -= process_size[i];
        }
 }
    printf("Worst Fit Allocation:\n");
```

```
    for (int i = 0; i < process_count; i++) {
        if (allocation[i] != -1)
            printf("Process %d -> Block %d\n", i + 1, allocation[i] + 1);
        else
            printf("Process %d -> Not Allocated\n", i + 1);
    }
}
int main() {
    int block_count, process_count;
    printf("Enter number of memory blocks: ");
    scanf("%d", &block_count);
    int block_size[block_count];
    printf("Enter size of each memory block:\n");
    for (int i = 0; i < block_count; i++) {
        scanf("%d", &block_size[i]);
    }
    printf("Enter number of processes: ");
    scanf("%d", &process_count);
    int process_size[process_count];
    printf("Enter size of each process:\n");
    for (int i = 0; i < process_count; i++) {
        scanf("%d", &process_size[i]);
    }
    int block_size_copy[block_count];    for (int i = 0; i < block_count; i++) block_size_copy[i]
= block_size[i];
    first_fit(block_size_copy, block_count, process_size, process_count);
    for (int i = 0; i < block_count; i++) block_size_copy[i] = block_size[i];
    best_fit(block_size_copy, block_count, process_size, process_count);
    for (int i = 0; i < block_count; i++) block_size_copy[i] = block_size[i];
    worst_fit(block_size_copy, block_count, process_size, process_count);
    return 0;
```

}

Result:

The program successfully demonstrates various memory allocation strategies (First Fit, Best Fit, and Worst Fit) and allocates memory to processes based on the selected strategy.

Output:

```
Enter size of each memory block:
100
500
200
300
600
Enter number of processes: 4
Enter size of each process:
212
417
112
426
First Fit Allocation:
Process 1 -> Block 2
Process 2 -> Block 5
Process 3 -> Block 2
Process 4 -> Not Allocated
Best Fit Allocation:
Process 1 -> Block 4
Process 2 -> Block 2
Process 3 -> Block 3
Process 4 -> Block 5
Worst Fit Allocation:
Process 1 -> Block 5
Process 2 -> Block 2
Process 3 -> Block 5
Process 4 -> Not Allocated
```