Os lab-8(23BRS1157)

MEMORY ALLOCATION - F2 SLOT (CH2024250101456)

1)Write a C/C++ program for dynamic memory allocation algorithm. Consider six memory partitions of size 150 KB, 500 KB, 250 KB, 300 KB, and 600 KB. In that order, these partitions need to be allocated to four processes of sizes 218 KB, 319 KB, 120 KB , 290KB, and 480KB. Calculate the external fragmentation caused if any.

Perform the allocation of processes using

1. **First Fit Algorithm**
2. **Best Fit Algorithm**
3. **Worst Fit Algorithm**

Code:

```
#include <stdio.h>

// Structure to represent a memory partition
typedef struct {     int size;
   int allocated; // 1 if allocated, 0 if free
   int processSize; // Size of the process allocated to this partition, if any
} MemoryPartition;

// Function to perform First Fit allocation
void firstFit(MemoryPartition partitions[], int numPartitions, int processSizes[], int numProcesses) {
int i, j;
   int externalFragmentation = 0;

   // Initialize all partitions as free
for (i = 0; i < numPartitions; i++) {
partitions[i].allocated = 0;
      partitions[i].processSize = 0;
   }

   printf("First Fit Allocation:\n");
```

```
   for (i = 0; i < numProcesses; i++) {
for (j = 0; j < numPartitions; j++) {
          if (partitions[j].allocated == 0 && partitions[j].size >= processSizes[i]) {
partitions[j].allocated = 1;
             partitions[j].processSize = processSizes[i];
             externalFragmentation += partitions[j].size - processSizes[i];
             printf("Process %d (%d KB) allocated to Partition %d (%d KB), Fragmentation: %d
KB\n",
                    i+1, processSizes[i], j+1, partitions[j].size, partitions[j].size - processSizes[i]);
break; // Move to the next process
          }
       }
       if (j == numPartitions) {
          printf("Process %d (%d KB) cannot be allocated.\n", i+1, processSizes[i]);
       }
   }
   printf("Total External Fragmentation (First Fit): %d KB\n\n", externalFragmentation); }

// Function to perform Best Fit allocation
void bestFit(MemoryPartition partitions[], int numPartitions, int processSizes[], int numProcesses) {
int i, j;
   int bestIndex, minWaste;     int
externalFragmentation = 0;

   // Initialize all partitions as free
for (i = 0; i < numPartitions; i++) {
partitions[i].allocated = 0;
partitions[i].processSize = 0;
   }

   printf("Best Fit Allocation:\n");
for (i = 0; i < numProcesses; i++) {
       bestIndex = -1;
       minWaste = 99999; // Initialize with a large value
for (j = 0; j < numPartitions; j++) {
          if (partitions[j].allocated == 0 && partitions[j].size >= processSizes[i]) {
int waste = partitions[j].size - processSizes[i];                if (waste < minWaste) {
minWaste = waste;
                bestIndex = j;
             }
          }
       }
       if (bestIndex!= -1) {
```

```
            partitions[bestIndex].allocated = 1;
            partitions[bestIndex].processSize = processSizes[i];
            externalFragmentation += partitions[bestIndex].size - processSizes[i];           printf("Process
%d (%d KB) allocated to Partition %d (%d KB), Fragmentation: %d KB\n",                    i+1,
processSizes[i], bestIndex+1, partitions[bestIndex].size, partitions[bestIndex].size - processSizes[i]);
        } else {
            printf("Process %d (%d KB) cannot be allocated.\n", i+1, processSizes[i]);
        }
    }
    printf("Total External Fragmentation (Best Fit): %d KB\n\n", externalFragmentation); }

// Function to perform Worst Fit allocation
void worstFit(MemoryPartition partitions[], int numPartitions, int processSizes[], int numProcesses)
{     int i,
j;
    int worstIndex;
    int externalFragmentation = 0;

    // Initialize all partitions as free
for (i = 0; i < numPartitions; i++) {
partitions[i].allocated = 0;
partitions[i].processSize = 0;
    }

    printf("Worst Fit Allocation:\n");
for (i = 0; i < numProcesses; i++) {
        worstIndex = -1;
int maxAvailable = 0;
        for (j = 0; j < numPartitions; j++) {
            if (partitions[j].allocated == 0 && partitions[j].size >= processSizes[i]) {
if (partitions[j].size > maxAvailable) {                    maxAvailable =
partitions[j].size;
                    worstIndex = j;
                }
            }
        }
        if (worstIndex!= -1) {
            partitions[worstIndex].allocated = 1;
            partitions[worstIndex].processSize = processSizes[i];
            externalFragmentation += partitions[worstIndex].size - processSizes[i];
```

```
            printf("Process %d (%d KB) allocated to Partition %d (%d KB), Fragmentation: %d KB\n",
i+1, processSizes[i], worstIndex+1, partitions[worstIndex].size, partitions[worstIndex].size -
processSizes[i]);
        } else {
            printf("Process %d (%d KB) cannot be allocated.\n", i+1, processSizes[i]);
        }
    }
    printf("Total External Fragmentation (Worst Fit): %d KB\n", externalFragmentation); }

int main() {
    // Define memory partitions
    int numPartitions = 5; // Corrected from 6 to 5 as per the sizes provided
    MemoryPartition partitions[] = {
        {150, 0, 0}, // Size in KB, Allocated, Process Size
        {500, 0, 0},
        {250, 0, 0},
        {300, 0, 0},
        {600, 0, 0}
    };

    // Define process sizes
    int numProcesses = 5; // Corrected from 4 to 5 as per the sizes provided
int processSizes[] = {218, 319, 120, 290, 480}; // Sizes in KB

    firstFit(partitions, numPartitions, processSizes, numProcesses);
bestFit(partitions, numPartitions, processSizes, numProcesses);
worstFit(partitions, numPartitions, processSizes, numProcesses);

    return 0;
}
```

Output:

```
ex5@208-51ThinkStation-P348:~/Desktop/work$ gcc first.c -o first
ex5@208-51ThinkStation-P348:~/Desktop/work$ ./first
First Fit Allocation:
Process 1 (218 KB) allocated to Partition 2 (500 KB), Fragmentation: 282 KB
Process 2 (319 KB) allocated to Partition 5 (600 KB), Fragmentation: 281 KB
Process 3 (120 KB) allocated to Partition 1 (150 KB), Fragmentation: 30 KB
Process 4 (290 KB) allocated to Partition 4 (300 KB), Fragmentation: 10 KB
Process 5 (480 KB) cannot be allocated.
Total External Fragmentation (First Fit): 603 KB

Best Fit Allocation:
Process 1 (218 KB) allocated to Partition 3 (250 KB), Fragmentation: 32 KB
Process 2 (319 KB) allocated to Partition 2 (500 KB), Fragmentation: 181 KB
Process 3 (120 KB) allocated to Partition 1 (150 KB), Fragmentation: 30 KB
Process 4 (290 KB) allocated to Partition 4 (300 KB), Fragmentation: 10 KB
Process 5 (480 KB) allocated to Partition 5 (600 KB), Fragmentation: 120 KB
Total External Fragmentation (Best Fit): 373 KB

Worst Fit Allocation:
Process 1 (218 KB) allocated to Partition 5 (600 KB), Fragmentation: 382 KB
Process 2 (319 KB) allocated to Partition 2 (500 KB), Fragmentation: 181 KB
Process 3 (120 KB) allocated to Partition 4 (300 KB), Fragmentation: 180 KB
Process 4 (290 KB) cannot be allocated.
Process 5 (480 KB) cannot be allocated.
Total External Fragmentation (Worst Fit): 743 KB
```