Write a C program to simulate the following contiguous memory allocation techniques.

a) Worst-fit

b) Best-fit

c) First-fit

```
#include <stdio.h>

void firstFit(int blockSize[], int blocks, int processSize[], int processes) {
   int allocation[processes];
   for(int i = 0; i < processes; i++)
      allocation[i] = -1;

   for(int i = 0; i < processes; i++) {
      for(int j = 0; j < blocks; j++) {
         if(blockSize[j] >= processSize[i]) {
            allocation[i] = j;
            blockSize[j] -= processSize[i];
            break;
         }
      }
   }

   printf("\nFirst-Fit Allocation:\n");
   for(int i = 0; i < processes; i++) {
      printf("Process %d -> ", i + 1);
      if(allocation[i] != -1)
         printf("Block %d\n", allocation[i] + 1);
      else
         printf("Not Allocated\n");
   }
}

void bestFit(int blockSize[], int blocks, int processSize[], int processes) {
   int allocation[processes];
   for(int i = 0; i < processes; i++)
      allocation[i] = -1;

   for(int i = 0; i < processes; i++) {
      int best = -1;
      for(int j = 0; j < blocks; j++) {
         if(blockSize[j] >= processSize[i]) {
            if(best == -1 || blockSize[j] < blockSize[best])
               best = j;
```

```
            }
        }
        if(best != -1) {
            allocation[i] = best;
            blockSize[best] -= processSize[i];
        }
    }

    printf("\nBest-Fit Allocation:\n");
    for(int i = 0; i < processes; i++) {
        printf("Process %d -> ", i + 1);
        if(allocation[i] != -1)
            printf("Block %d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

void worstFit(int blockSize[], int blocks, int processSize[], int processes) {
    int allocation[processes];
    for(int i = 0; i < processes; i++)
        allocation[i] = -1;

    for(int i = 0; i < processes; i++) {
        int worst = -1;
        for(int j = 0; j < blocks; j++) {
            if(blockSize[j] >= processSize[i]) {
                if(worst == -1 || blockSize[j] > blockSize[worst])
                    worst = j;
            }
        }
        if(worst != -1) {
            allocation[i] = worst;
            blockSize[worst] -= processSize[i];
        }
    }

    printf("\nWorst-Fit Allocation:\n");
    for(int i = 0; i < processes; i++) {
        printf("Process %d -> ", i + 1);
        if(allocation[i] != -1)
            printf("Block %d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

int main() {
```

```
    int blockSize[10], processSize[10], blocks, processes;

    printf("Enter number of memory blocks: ");
    scanf("%d", &blocks);
    printf("Enter sizes of %d blocks:\n", blocks);
    for(int i = 0; i < blocks; i++)
        scanf("%d", &blockSize[i]);

    printf("Enter number of processes: ");
    scanf("%d", &processes);
    printf("Enter sizes of %d processes:\n", processes);
    for(int i = 0; i < processes; i++)
        scanf("%d", &processSize[i]);

    int blockCopy1[10], blockCopy2[10], blockCopy3[10];
    for(int i = 0; i < blocks; i++) {
        blockCopy1[i] = blockSize[i];
        blockCopy2[i] = blockSize[i];
        blockCopy3[i] = blockSize[i];
    }

    firstFit(blockCopy1, blocks, processSize, processes);
    bestFit(blockCopy2, blocks, processSize, processes);
    worstFit(blockCopy3, blocks, processSize, processes);

    return 0;
}
```

Output

```
Enter number of memory blocks: 2
Enter sizes of 2 blocks:
3
6
Enter number of processes: 2
Enter sizes of 2 processes:
6
8

First-Fit Allocation:
Process 1 -> Block 2
Process 2 -> Not Allocated

Best-Fit Allocation:
Process 1 -> Block 2
Process 2 -> Not Allocated

Worst-Fit Allocation:
Process 1 -> Block 2
Process 2 -> Not Allocated


=== Code Execution Successful ===
```