Write a c program to implement the following contiguous memory locations:

1. First-fit
2. Best-fit
3. Worst-fit

```
#include <stdio.h>

#define MAX 100

void printAllocation(char* strategy, int allocation[], int processSize[], int processCount, int blockSnapshot[]) {
   printf("\n\n%s Allocation:\n", strategy);
   printf("File No.\tFile Size\tBlock No.\tBlock Size\n");
   for (int i = 0; i < processCount; i++) {
      if (allocation[i] != -1)
         printf("%d\t\t%d\t\t%d\t\t%d\n", i + 1, processSize[i], allocation[i] + 1, blockSnapshot[allocation[i]]);
      else
         printf("%d\t\t%d\t\tNot Allocated\t--\n", i + 1, processSize[i]);
   }
}

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

```
    printAllocation("First Fit", allocation, processSize, processes, blockSnapshot);
}
void bestFit(int blockSize[], int blocks, int processSize[], int processes) {
    int allocation[MAX], blockSnapshot[MAX];
    for (int i = 0; i < processes; i++) allocation[i] = -1;
    for (int i = 0; i < blocks; i++) blockSnapshot[i] = blockSize[i];

    for (int i = 0; i < processes; i++) {
        int bestIdx = -1;
        for (int j = 0; j < blocks; 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];
        }
    }
    printAllocation("Best Fit", allocation, processSize, processes, blockSnapshot);
}

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

    for (int i = 0; i < processes; i++) {
        int worstIdx = -1;
```

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

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

    printAllocation("Worst Fit", allocation, processSize, processes, blockSnapshot);
}

int main() {
    int blockSize[MAX], processSize[MAX];
    int blocks, processes;

    printf("Enter number of memory blocks: ");
    scanf("%d", &blocks);
    printf("Enter size of each memory block:\n");
    for (int i = 0; i < blocks; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &blockSize[i]);
    }

    printf("Enter number of files (processes): ");
    scanf("%d", &processes);
    printf("Enter size of each file:\n");
```

```
    for (int i = 0; i < processes; i++) {
        printf("File %d: ", i + 1);
        scanf("%d", &processSize[i]);
    }

    int tempBlock1[MAX], tempBlock2[MAX], tempBlock3[MAX];
    for (int i = 0; i < blocks; i++) {
        tempBlock1[i] = tempBlock2[i] = tempBlock3[i] = blockSize[i];
    }

    firstFit(tempBlock1, blocks, processSize, processes);
    bestFit(tempBlock2, blocks, processSize, processes);
    worstFit(tempBlock3, blocks, processSize, processes);

    return 0;
}
```

```
Enter number of memory blocks: 3
Enter size of each memory block:
Block 1: 300
Block 2: 400
Block 3: 100
Enter number of files (processes): 2
Enter size of each file:
File 1: 213
File 2: 241


First Fit Allocation:
File No.        File Size       Block No.       Block Size
1               213             1               300
2               241             2               400


Best Fit Allocation:
File No.        File Size       Block No.       Block Size
1               213             1               300
2               241             2               400


Worst Fit Allocation:
File No.        File Size       Block No.       Block Size
1               213             2               400
2               241             1               300
```