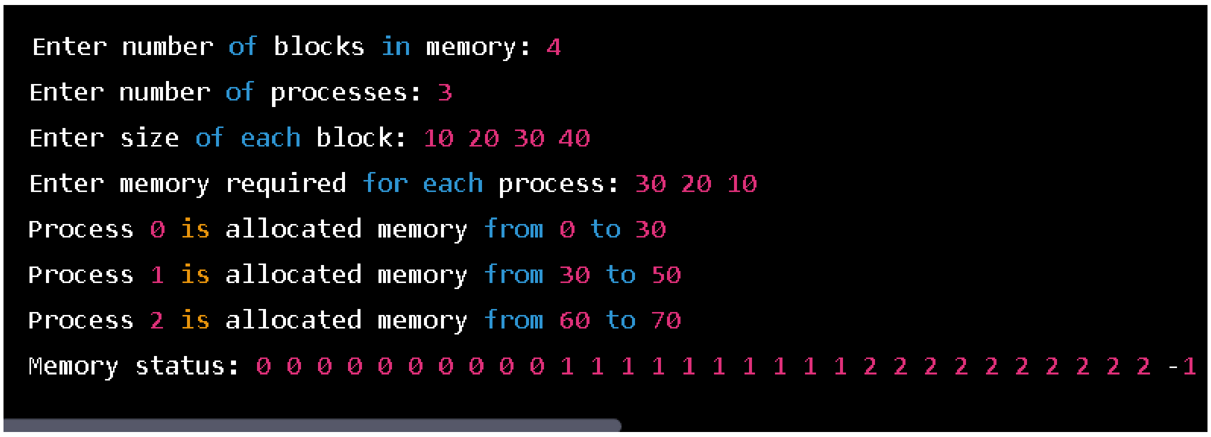
```
#include <stdio.h>

#define MAX_BLOCKS 100
#define MAX_PROCESSES 50
#define MAX_MEMORY 100

int memory[MAX_MEMORY]; // Representation of memory
int blocks[MAX_BLOCKS]; // Representation of blocks
int processes[MAX_PROCESSES]; // Representation of processes
int n, m; // n is number of blocks and m is number of processes

// Function to initialize memory
void init_memory() {
  int i;
  for (i = 0; i < MAX_MEMORY; i++) {
    memory[i] = -1;
  }
}

// Function to allocate memory to a process
void allocate_memory(int process_id, int size) {
  int i, j;
  for (i = 0; i < n; i++) {
    int block_start = blocks[i];
    int block_end = block_start + size;
    int flag = 0;
    if (block_end > MAX_MEMORY) {
      continue;
    }
    for (j = block_start; j < block_end; j++) {
      if (memory[j] != -1) {
        flag = 1;
        break;
      }
    }
    if (flag == 0) {
      for (j = block_start; j < block_end; j++) {
        memory[j] = process_id;
      }
      printf("Process %d is allocated memory from %d to %d\n", process_id, block_start,
block_end);
      return;
    }
  }
  printf("Memory is not sufficient to allocate to process %d\n", process_id);
}

// Function to display the memory status
void display_memory() {
  int i;
  printf("Memory status: ");
  for (i = 0; i < MAX_MEMORY; i++) {
    printf("%d ", memory[i]);
  }
  printf("\n");
}

int main() {
  int i;
  printf("Enter number of blocks in memory: ");
  scanf("%d", &n);
  printf("Enter number of processes: ");
  scanf("%d", &m);
  printf("Enter size of each block: ");
  for (i = 0; i < n; i++) {
    scanf("%d", &blocks[i]);
  }
  printf("Enter memory required for each process: ");
  for (i = 0; i < m; i++) {
    scanf("%d", &processes[i]);
  }
  init_memory();
  for (i = 0; i < m; i++) {
    allocate_memory(i, processes[i]);
  }
  display_memory();
  return 0;
}
```



```
Enter number of blocks in memory: 4
Enter number of processes: 3
Enter size of each block: 10 20 30 40
Enter memory required for each process: 30 20 10
Process 0 is allocated memory from 0 to 30
Process 1 is allocated memory from 30 to 50
Process 2 is allocated memory from 60 to 70
Memory status: 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 -1
```