```c
#include <stdbool.h>
#include <stdio.h>

#define TOTAL_MEMORY 100
#define MFT_BLOCKS 50

void printMemory(int mft[], int mvt[], int mft_size, int mvt_size) {
  printf("MFT Memory: ");
  for (int i = 0; i < mft_size; i++) {
    printf("%d ", mft[i]);
  }
  printf("\n");

  printf("MVT Memory: ");
  for (int i = 0; i < mvt_size; i++) {
    if (mvt[i] == -1) {
      printf(". ");
    } else {
      printf("%d ", mvt[i]);
    }
  }
  printf("\n");
}

int allocateMFT(int mft[], int task_size, int mft_size) {
  for (int i = 0; i <= mft_size - task_size; i++) {
    bool fit = true;
    for (int j = i; j < i + task_size; j++) {
      if (mft[j] != 0) {
        fit = false;
        break;
      }
    }
    if (fit) {
      for (int j = i; j < i + task_size; j++) {
        mft[j] = 1;
      }
      return i;
    }
  }
  return -1;
}

void deallocateMFT(int mft[], int start, int task_size) {
  for (int i = start; i < start + task_size; i++) {
    mft[i] = 0;
  }
}

int allocateMVT(int mvt[], int task_size, int mvt_size) {
  for (int i = 0; i <= mvt_size - task_size; i++) {
    bool fit = true;
    for (int j = i; j < i + task_size; j++) {
      if (mvt[j] != -1) {
        fit = false;
        break;
      }
    }
    if (fit) {
      for (int j = i; j < i + task_size; j++) {
        mvt[j] = 1;
      }
      return i;
    }
  }
  return -1;
}

void deallocateMVT(int mvt[], int start, int task_size) {
  for (int i = start; i < start + task_size; i++) {
    mvt[i] = -1;
  }
}

int main() {
  int mft[MFT_BLOCKS] = {0};
```

```
  int mvt[TOTAL_MEMORY] = {-1};

  printf("Initial State:\n");
  printMemory(mft, mvt, MFT_BLOCKS, TOTAL_MEMORY);

  printf("\nAllocating tasks:\n");
  int task_size = 10;
  printf("Allocating task of size %d in MFT\n", task_size);
  int start = allocateMFT(mft, task_size, MFT_BLOCKS);
  if (start != -1) {
    printf("Task allocated at index %d\n", start);
  } else {
    printf("Failed to allocate task in MFT\n");
  }

  printf("Allocating task of size %d in MVT\n", task_size);
  start = allocateMVT(mvt, task_size, TOTAL_MEMORY);
  if (start != -1) {
    printf("Task allocated at index %d\n", start);
  } else {
    printf("Failed to allocate task in MVT\n");
  }

  printf("\nCurrent State:\n");
  printMemory(mft, mvt, MFT_BLOCKS, TOTAL_MEMORY);

  printf("\nDeallocating tasks:\n");
  printf("Deallocating task of size %d from MFT starting at index %d\n",
         task_size, 0);
  deallocateMFT(mft, 0, task_size);

  printf("Deallocating task of size %d from MVT starting at index %d\n",
         task_size, 0);
  deallocateMVT(mvt, 0, task_size);

  printf("\nFinal State:\n");
  printMemory(mft, mvt, MFT_BLOCKS, TOTAL_MEMORY);

  return 0;
}
```