```
/* Helper functions for Node 1 - these are not part of MRAPI, and could be
   implemented using various appropriate mechanisms */

/* Function which uses some mechanism (e.g. MCAPI) to send a message to Node
2 */
void send_to_node2(int);

/* Function via which Node 1 waits for Node 2 to complete its work */
void wait_for_notification_from_node2();

void* START_OF_HEAP;
int SIZE_OF_HEAP;

/* Code for Node 1 functionality */
int node1_remote_memory_use_case_1(Entity * entities_to_be_processed,
            float * scores_to_be_computed, unsigned int number_of_entities)

      /*
       'entities_to_be_processed' is a linked list of 'Entity' structures,
which are going to be processed by Node 2.
       Since the 'Entity' data is not contiguous, but elements of the list
may reside close together in memory, software
       caching is an appropriate access mechanism for the remote access.

       'scores_to_be_computed' is an array which is to be filled, by Node 2,
with a score for each entity.  Since
       elements of this array are contiguous, DMA is an appropriate access
mechanism for the remote access.

       */

{

      mrapi_status_t status; /* For error checking */

      /* Handles for software cache- and DMA-accessed remote memory */
      mrapi_rmem_hndl_t sw_cache_hndl;
      mrapi_rmem_hndl_t dma_hndl;

      /* We want Node 2 to process the linked list
'entities_to_be_processed'.  But elements of this list can be
         located anywhere on the heap, thus we need to make the heap
available remotely.
       */
      sw_cache_hndl = mrapi_rmem_create(AGREED_ID_FOR_SW_CACHE,
                                        START_OF_HEAP,
                                        MRAPI_ACCESS_TYPE_SW_CACHE,
                                        NULL,
                                        SIZE_OF_HEAP,
                                        &status);

      // CHECK STATUS FOR ERROR
      if (status != MRAPI_SUCCESS) {
```