```
      /*
      Node 1 is "looking after" Node 2, and waits for Node 2 to send
      messages requesting more data.  On receiving such a message, Node 1
      allocates some more memory which it makes available to Node 2.
      Once Node 2 signals that it has completed, Node 1 deletes all the
      allocated memory.
       */

{

      mrapi_status_t status; /* For error checking */

      int message;

      int next_buf = 0;

      while (wait_for_message_from_node_2(&message))
      {
            if(message == QUIT)
            {
                  /* Node 2 says "I'm done", so we can exit the loop */
                  break;
            }

            assert(message == MORE_MEMORY_PLEASE);

            /* Node 2 needs some more memory, and will have sent another
message saying how much */

            int amount_of_data_required_in_bytes;

            wait_for_message(&amount_of_data_required_in_bytes);

            /* Allocate the desired amount of memory locally */
            buffers[next_buf].pointer = (char*) malloc(
             amount_of_data_required_in_bytes * sizeof(char) );

            /* We want to make this memory available remotely, so obtain an
id for the new piece of remote memory */
            mrapi_rmem_id_t id = get_fresh_rmem_id();

            /* Now promote the freshly allocated buffer to be visible
remotely */
            buffers[next_buf].handle = mrapi_rmem_create(
                  id,
                  buffers[next_buf].pointer,
                  AGREED_ACCESS_TYPE,
                  NULL,
                  amount_of_data_required_in_bytes * sizeof(char),
                  &status);

            // CHECK status FOR ERRORS - OMITTED
```