Exercise 9-1  HashMap macros

a) Write a declarative Rust macro that allows creating a HashMap from keys and values, freeing the programmer from manually allocating the empty data structure and inserting one element after the other. The macro should have an interface resembling the vec! macro. Make sure that the macro is fully hygienic, i.e. all function calls within the macro should be properly qualified.

Test your macro with the following invocations:

```rust
let m1: HashMap<i32, String> = hmap![];
let m2 = hmap!["key1" => 1, "key2" => 2, "key3" => 3];
```

b) Extend the previous solution such that the HashMap is allocated with an explicitly specified capacity, i.e. instead of HashMap::new() the function HashMap::with_capacity() should be used. The initial capacity of the HashMap should correspond to the given number of arguments to the macro invocation.

**Hint:** For counting the arguments a TT muncher can be used.