COMPUTER SCIENCE TRIPOS Part IB – 2021 – Paper 4

2 Programming in C and C++ (djg11)

(*a*) A key/value store holds pairs of strings in a non-volatile memory (NVM). The NVM is mapped as a memory region between two hardcoded virtual addresses, NV_START and NV_END. This memory is all set to zero as manufactured and can be freely read, but any write operations result in a logical ORing of the new data with the old data at the addressed location. Bits can never be cleared. Strings representing keys and values will be fewer than 250 characters long. Eventually the memory will fill up, but sufficient is available for the intended application.

The API provides the following two operations:

```
int store(char *key, char *value) // returns non-zero on error, and
char *lookup(char *key)           // returns NULL if not found.
```

Provide a full C implementation of the `store` routine, explaining how the lookup function and changes to values under a given key would work, if this is not obvious from your code. Code efficiency is not important.
Hint: You can directly access the non-volatile store using a construct such as `((char *)NV_START)[offset]`. [10 marks]

(*b*) The following text *almost* constitutes both a C++ and Java program:

```
class Foo { public: int x, y; };
class Test {
  private: void f1(Foo p) { p.x = 1; }
  private: void f2(Foo &q) { q.y = 2; }  //[Java]: ignore '&'
  public: void test() {
    Foo p;  //[Java]: replace with: 'Foo p = new Foo();'
    p.x = p.y = 99;
    f1(p);
    Foo q = p;
    p = q;
    f2(q);
    print("HERE");
  }
};
```

(*i*) Explain the storage structure accessible from variables `x` and `y` when control reaches `print("HERE")` following a call to `test()`, both in the Java and the C++ interpretations. [3 marks]

(*ii*) For the C++ interpretation, show how adjustments *only to the definition* of `class Foo` can cause (useful) debugging-style output to be produced at *as many places as possible* during the execution of method `test()`. [7 marks]