ECE326 – Fall 2019: Week 8 Exercise Questions

1. True or False [1 mark each]

Circle T is true, otherwise circle F for false.

1. C macro functions always produce syntactically correct C/C++ code. **T** [✓ **F**]
2. Once a macro is used, it cannot be used during the same macro expansion. [✓ **T**] **F**
3. To prevent infinite recursion, it is not possible for concatenation to form a token that can be used as a macro. **T** [✓ **F**]
4. The main benefit of optional compilation over inheritance is performance (speed). **T** [✓ **F**]

Executable size is the main benefit. Optional compilation does not significantly improve performance except possibly due to less cache miss. Code organization can be another benefit.

5. Clever use of C preprocessor macro allows for static introspection. **T** [✓ **F**]

Preprocessor have no knowledge of host language.

6. A constexpr function or variable is exclusively for compile-time use. **T** [✓ **F**]

2. Generic Programming **[10 marks]**

Use C macros instead of template programming to generate a Stack class of generic element type. Your stack should be type safe and should have the pop and push member functions. Assume input will be an unadorned type (not a pointer, not a reference, not const or other qualifiers).

```
// creates a class named IntStack with integer elements
DEFINE_STACK(IntStack, int);

// creates a class named ComplexStack with Complex elements
DEFINE_STACK(ComplexStack, Complex);
```

see macro.cpp

3. Compile-Time String **[10 marks]**

Write a compile-time class, ConstStr, which provides the following three compile-time methods: 1) hash(), which returns a djb2 hash of the string (http://www.cse.yorku.ca/~oz/hash.html), 2) startswith(substr), which only returns true if the string starts with the substring substr, and 3) endswith(substr), which only returns true if the string ends with the substring substr.

see conststr.cpp

4. C Macro **[5 marks]**

The PP_NARG macro can count the number of arguments that is passed into the maco function. However, it doesn't work correctly if no arguments are passed in. Fix the macro so that it also supports no arguments.

```
#define PP_NARG(...) PP_NARG_(__VA_ARGS__, PP_RSEQ_N())
#define PP_NARG_(...) PP_ARG_N(__VA_ARGS__)
#define PP_ARG_N( _1, _2, _3, _4, _5, _6, _7, _8, _9, N, ...) N
#define PP_RSEQ_N() 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
```

see macro.cpp