In class Practice 12/06/2019 No submission required for this practice

Part I Exception handling

1. What’s a “resource leak”?
2. If no exceptions are thrown in a try block, where does control proceed to after the try block completes execution?
3. What happens if an exception is thrown outside a try block?
4. If an exception is thrown in a function, say, f(), but not handled there, the exception is propagated to the function that called f().
5. What happens if several handlers match the type of the thrown object?
6. Give a key advantage and a key disadvantage of using catch(...).
7. Must throwing an exception cause program termination?
8. Each member-function definition outside its corresponding class template definition must begin with template and the same template parameters as its class template.

Part II template

1. To instantiate and call, a template function requires special syntax.
2. The template prefix can be written template <typename identifier> or template <class identifier> with the same results.
3. All template definitions begin with the keyword________, followed by a list of template parameters enclosed in ___________.
4. Class templates also are called______________ types.
5. It is possible to have more than one type parameter in a template definition.
6. In implementing class template member functions, the functions are themselves templates.
7. In the template prefix, template<class T>, what kinds of variables is the parameter T?
   a. T must be a type built into C++ such as int or double.
   b. T must *not* be a class.
   c. T *must* be a class.
   d. T can be any type, whether built into C++ or programmer defined.
   e. T can be any type, whether built into C++ or programmer defined, but subject to restrictions. Explain what these are.

Part III debug and program

1. Correct the errors if there are any

```
#include <iostream>
template<class T>
class A
{
public:
  A(){};
  void B(){std::cout << "I am B"<<std::endl <<"x is " <<x<<std::endl;}
  <T> getX() {return x;}
  void setX (T theValue) { x = theValue;}
private:
  <T> x;
};

int main()
{
    A<int> a;
    a.B();
}
```

2. In C++, an exception object can be a user-defined type or any type that is built-into C++ language. Write a program which will ask user to enter number of cups of coffee and number of cookies. Your program will compute number of cookies per coffee based on user’s input. Because number of coffee can be 0 so divided by zero may occur. Instead of creating exception object like the textbook, you will develop a class call NoCoffee. When there is no coffee, your program will throw a NoCoffee exception object saying there is no coffee.