CS3: Programming Patterns

QUIZ #1

1. What does this code output?

```
int a[] = {10, 20, 30, 40};
int s=0;
for(auto e: a) s += e;
cout << s;
```

   a) 10
   b) 40
   c) 100
   d) 200
   e) this code is incorrect and would not compile

2. The below prototyped function

```
void myFunc(int a, int b=0, int c=1);
```

   may NOT be invoked as:

   a) myFunc(1,2,3);
   b) myFunc(1,2);
   c) myFunc(1);
   d) myFunc();
   e) the function prototype is illegal in C++

3. What does the following code print:

```
int a = 1;
int &b = a;
++a;
++b;
cout << b;
```

   a) 1
   b) 2
   c) 3
   d) 4
   e) this code is illegal and will not compile

4. Consider the following function prototype and variable declaration:

```
int *func();
int v = 55;
```

   What is a correct function invocation

```
a)     func() =  v;
b)    *func() =  v;
c)     func() = *v;
d)    *func() = *v;
e)    &func() =  v;
```

5. Consider the following three statements:

```
int *ptr1 = 0;
int *ptr2 = NULL;
int *ptr3 = nullptr;
```

   Which statement is preferred and why?

   a) first, because there is no confusion as to how the pointer is initialized
   b) second, because pointers are initialized with NULL value
   c) third, because it is a newer construct in C++
   d) third, because "nullptr" is of type pointer which eliminates type ambiguity
   e) all of the above are equally recommended for use

6. What is the output of the following portion of code?

```
int a[10] = {0};
cout << a[9];
```

   a) -1
   b) 0
   c) arbitrary integer
   d) NULL
   e) this code is illegal and will not compile

7. What does the following code print?

```
int myFunc(){
  static int a=0;
  return ++a;
}

int main(){
  cout << myFunc() << myFunc();
}
```

THIS QUESTION HAS MULTIPLE ANSWERS SINCE THE ORDER OF FUNCTION EXECUTION IN EXTRACTION OPERATOR IS UNDEFINED. THE QUESTION WAS MEANT TO TEST THE KNOWLEDGE OF "static".

   a) 11
   b) 12
   c) 21
   d) 22
   e) this code is illegal and will not compile

8. The process of determining the type parameter of a standalone function template on the basis of the type of arguments is called:

   a) instantiation
   b) deduction
   c) definition
   d) generic programming
   e) such determination can only be done for class templates

9. Consider the following class definition

```
template <typename T>
class MyClass {
public:
    T myfunc();
private:
    T a_;
};
```

   What would be a correct out-of-line definition of function myfunc()

```
a)  template<typename T> T  MyClass<T>::myfunc(){ return a_;}
b)  template<typename T>    MyClass<T>::myfunc(){ return a_;}
c)  template<typename T> T  MyClass   ::myfunc(){ return a_;}
d)                       T  MyClass<T>::myfunc(){ return a_;}
e)  template<typename T> T              myfunc(){ return a_;}
```

10. Consider the following templated class definition

```
   template <typename T=int, int Size=10>
   class MyClass{
     // details of class definition
   };
```

Which of object declarations below is NOT syntactically legal

```
a) MyClass<double,100> myobj;
b) MyClass<double>     myobj;
c) MyClass<>           myobj;
d) MyClass             myobj;
e) templated class cannot be instantiated in C++
```