Question	Points	Score
Part 1	30	
16	4	
17	6	
18	10	
19	5	
20	5	
21	15	
22	15	
23	15	
24	15	
Total	120	

Part I - Multiple Choice

[30 marks]

For the following questions, please circle (or indicate as specified by the question) your answer directly on the exam sheet. Note that some questions are worth more than one point.

1. [1 point] The following is(are) true about C++
   A. It is a dynamically typed language
   B. It allows dynamic dispatch of functions
   C. It does not allow inheritance
   D. It performs garbage collection
   E. More than one of the above statements are true

2. [1 point] When writing C++ code, the programmer, not the compiler, is responsible for:

   1. Ensuring no uninitialized values are used
   2. Ensuring there is the correct amount of space allocated for each stack frame
   3. Ensuring memory allocated on the heap is reclaimed
   4. Ensuring pointer addresses used are correct

   Which of the following correctly combines only the correct statements from the numbered list above.
   A. Statements 1 and 4 are correct
   B. Statement 2 is correct
   C. Statements 1, 3, and 4 are correct
   D. Statements 2, 3 and 4 are correct

3. [1 point] The syntax: const int * bob; indicates that bob is a:
   A. Regular pointer to regular int type
   B. Constant pointer to regular int type
   C. Regular pointer to constant int type
   D. Constant pointer to constant int type

4. [1 point] Given the following code segment, what is the correct call sequence of constructors and destructors.

```
1 class A{};
2 class B :public A{};
3
4 int main (){
5     A a1;
6     A * a2 = new B();
7     delete a2;
8 }
```

A. B default constructor, B destructor

B. Because no constructors and destructors are defined for the classes, the code will not compile

C. A default constructor, A default constructor, B default constructor, B destructor, A destructor, A destructor

D. A default constructor, B default constructor, B destructor, A destructor

E. The above code will cause a memory error and crash

5. [4 points] Consider the following function:

```
1 template <class T>
2 T average(T *atArray, int nNumValues) {
3           T tSum = 0;
4           for (int nCount=0; nCount < nNumValues; nCount++)
5              { tSum += atArray[nCount]; }
6           tSum = tSum / nNumValues;
7           return tSum;
8 }
```

Which of the following statements about the class/type T must be true in order for the code to compile and run without crashing?

It must be some kind of numeric type	T	F
Must have < operator defined	T	F
Must have the [ ] access operator defined	T	F
Must have copy constructor and assignment operator	T	F

6. [8 points] The perceived type of the `this` special variable within the scope of a function will change based on various modifiers found in the current function's defined header.

```
class A  {
    public:
        int a(int i);
        static int b(int i);
        const int c(int i);
        int d(int i) const;
};

int e(int i);
```

Available `this` types:

1. `A const`
2. `A* const`
3. `A*`
4. `const A* const`
5. `null`
6. Does not apply (no `this` type will be available in this function's scope)

(a) Based on the code above, match the perceived type of `this` below for each function a-e by writing the associated number from the list of available types.

A. Type of `this` in `a`: _____

B. Type of `this` in `b`: _____

C. Type of `this` in `c`: _____

D. Type of `this` in `d`: _____

E. Type of `this` in `e`: _____

(b) Within the function `a` above, what are we allowed to do with the `this` object in the scope of that function? CIRCLE ANY/ALL THAT APPLY

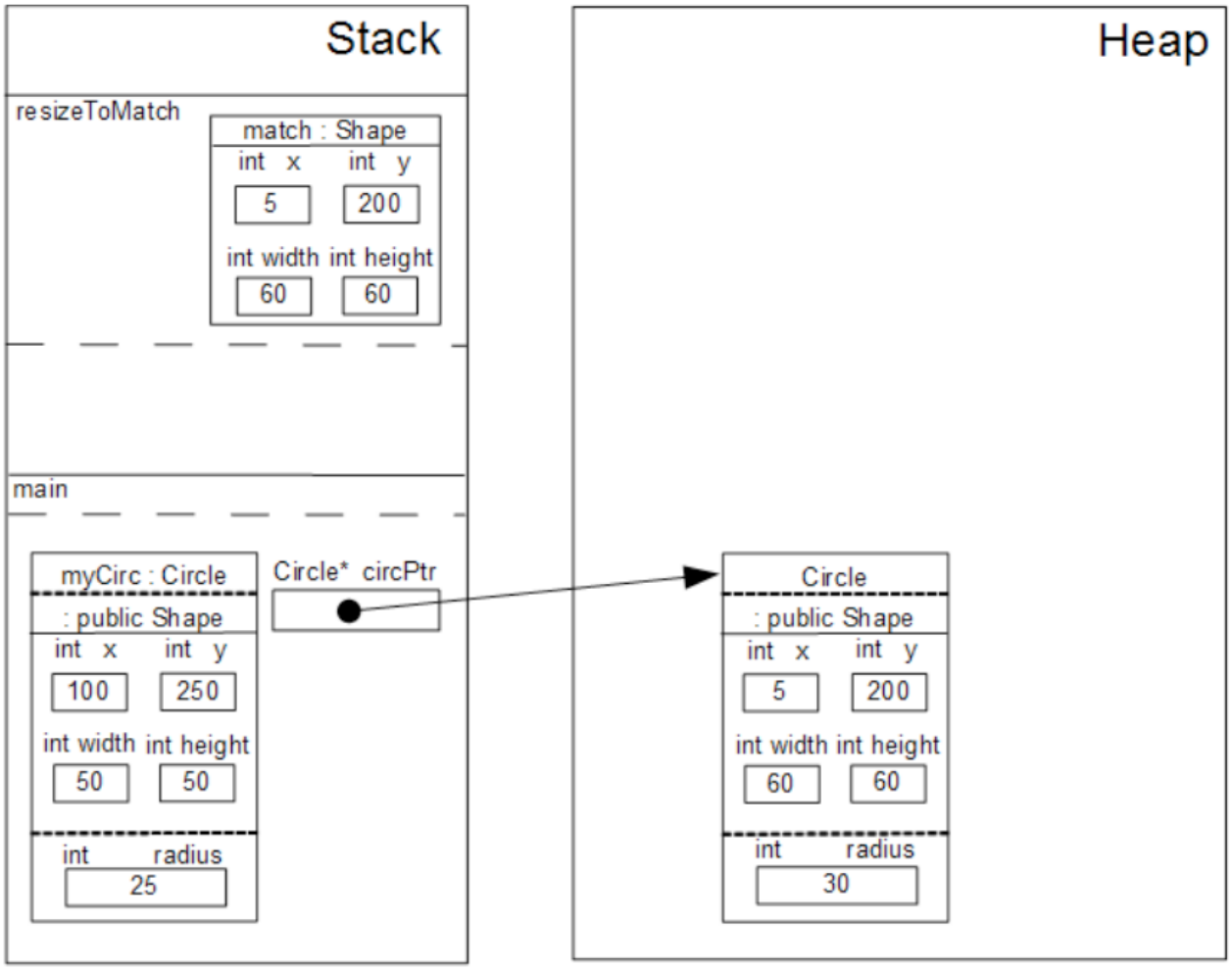
A. Call const functions
B. Change private member data
C. Return a non-const pointer to it
D. Return a const pointer to it

(c) Within the function `d` above, what are we allowed to do with the `this` object in the scope of that function? CIRCLE ANY/ALL THAT APPLY

A. Call const functions
B. Change private member data
C. Return a non-const pointer to it
D. Return a const pointer to it

(d) Within the function `e` above, what are we allowed to do with the `this` object in the scope of that function? CIRCLE ANY/ALL THAT APPLY

A. Call const functions
B. Change private member data
C. Return a non-const pointer to it
D. Return a const pointer to it



The state of memory is shown just as we enter scope of the `resizeToMatch` function. The function's intent is to change the dimensions of the current `Shape` to match those of the parameter `Shape`. The function is a member function of the `Shape` class, and Circle is a subclass of `Shape`. The `Circle` subclass overrides the `Shape resizeToMatch` function to also update its radius attribute.

7. [1 point] Which of the following function calls is a valid way to call the resizeToMatch function.

   A. `(*circPtr).resizeToMatch(circPtr)`

   B. `circPtr->resizeToMatch(circPtr)`

   C. `myCirc.resizeToMatch(*circPtr)`

   D. `myCirc->resizeToMatch(*circPtr)`

8. [6 points] Assume resizeToMatch was called on the myCirc Circle object. Which of the following statements is/are true

The `resizeToMatch` function's `Shape` parameter is passed by value	T	F
Slicing has occurred	T	F
The `resizeToMatch` function will not be able to change the radius of `this`	T	F
The `resizeToMatch` function will not be able to access the radius of `match`	T	F
The `circPtr` is considered a safe pointer	T	F
The `Shape` and `Circle` class conform to the Liskov Substitution Principle	T	F

9. [1 point] Which of the following statement(s) about User Stories is/are correct?
    A. Users Stories are generating in a meeting with the Client
    B. They encourage deferring detail until later
    C. They are intended to avoid "big design up front"
    D. They should be independent and estimable
    E. All of the above statements are correct

10. [1 point] Which of the following statements about the Decorator pattern are true?
    A. It can be used to add and remove behaviour dynamically at run time
    B. It is still viewed logically as a single unit, regardless of the number of decorator layers applied
    C. It works by wrapping one object inside another with the same interface
    D. All of the above statements are correct

11. [1 point] Creational Design Patterns
    A. Are all based on the Abstract Factory pattern
    B. Are used to encapsulated the creation of objects
    C. Are only useful when constructors are statically dispatched
    D. Two of the above statements are correct

12. [1 point] Why does the Interface Segregation Principle aim to avoid fat interfaces?
    A. High level modules should not depend on lower level modules, because higher level interfaces are less likely to change.
    B. Because excessive pollution of the interface prevents derived classes from being used as if they were the interface class.
    C. While fat interfaces reduce coupling between the subclasses and the interface, the large interface becomes difficult to use.
    D. Any class inheriting the interface, that should be instantiated as an object, must provide an implementation for all interface functions, which may be empty or meaningless.
    E. None of the above statements are correct

13. [1 point] The Adapter pattern:

    A. Can be implemented in C++ using multiple inheritance

    B. Uses an adapter object that mimics the interface of the target object the client wishes to use

    C. Uses an adapter object to translate requests to an existing interface

    D. Both a) and c) are true

    E. All of the above are true

14. [1 point] The Builder pattern is used to:

    A. Create objects by merging classes with each other

    B. Control the structure of complex, hierarchical class relationships, by building up from the bottom

    C. Create objects through a similar process of steps, but allow the actual representation to vary

    D. Control the creation of objects where there are many constructors with long and complex parameter lists

15. [1 point] You have created a mobile application which gives the user many options in how their calendar data is displayed. The ability to combine and layer different display options is the main selling feature of your software. Switching display options around is efficient and fast, and you are able to add new display options to your software without difficulty. It is likely that you have wisely chosen the:

    A. Adapter Pattern

    B. Command Pattern

    C. Facade Pattern

    D. Decorator Pattern

Part II - Written Answer

- *For the following questions, write your answer in the space provided, using diagrams as appropriate.*
- *You do NOT need to write full sentences if you don't want to. You can use point form.*
- *Commenting your code can make it clearer what you are trying to do, and may result in part marks if the code itself isn't correct.*

Short Answer [30 marks]

16. [4 points] What type of error might you get if you omitted the include guard (example below) in your C++ header files and why?

```
1 #ifndef SOMESOURCEFILE_H
2 #define SOMESOURCEFILE_H
3 ...//header content
4 #endif
```

17. (a) [4 points] Give a brief explanation of the difference between static and dynamic dispatch.

(b) [2 points] How do you indicate to the compiler which type of dispatch is to be used?

18. (a) [6 points] Draw a UML class diagram from the class definitions here, as written, listing all functions and attributes (show as associations where appropriate). You DO NOT have to list parameters, return types or attribute types.

```
class UserData {
   public:
    UserData* getInstance();
    addUser( Profile user);
    removeUser( Profile user);
    addScore (Profile user, int score);
   private:
    UserData();
    static UserData * instance;
    vector<Profile> users;
};
```

```
class Profile {
   public:
    Profile();
    virtual ~Profile();
    void addScore(int score);
   private:
    string name;
    string password;
    vector<int> scores;
};
class LocalProfile : public Profile {
   public:
    LocalProfile();
    ~LocalProfile();
    void addScore(int score);
   private:
    int highScore;
};
```

(b) [4 points] From the example given on the previous page, identify a creational design pattern being used by this system, and list its identifying features

19. [5 points] There is a considerable memory leak in your software system. You suspect after hours of debugging that something is wrong with the following object usage. Here `TimedSession` inherits from `Session`, i.e. `class TimedSession: public Session;`

```
1 Session * currentSession = new TimedSession(2000);
2 ...
3 delete currentSession;
```

Is it possible, when using the object in this way, for it to cause a memory leak? If so, what is the likely cause and why would that cause the leak? If not, why is it not possible?

20. [5 points] Consider the following class:

```
class EmailMessage {
    public:
        void send();
        void setFrom(const string& address);
        void setTo(const string& address);
        void setSubject(const string& subject);
        void setBody(const string& body);
        void login(const string& username, const string& password);
        void updateInbox();
};
```

Which SOLID principle does this class violate and why? How should it be changed to adhere to that principle?

Long Answer [60 marks]

21. [15 points] Consider the following class declaration

```
1 class B : public A {
2     public:
3         B();                // 3 marks
4         B(std::string in);  // 4 marks
5         B(const B& right);  // 5 marks
6         ~B();               // 3 marks
7        p    rivate:
8         std::string * b;
9 };
```

Provide an implementation for all of the above public methods. You may assume `A` has been correctly implemented, and includes a default constructor, as well as the big three.

```
/** Default constructor
  * Ensures all private members are initialized
  */
```

```
/** One-Parameter constructor
 *  b is initialized with the same content as the input string 'in'
 */
```

```
/** Copy constructor
 * The copy constructor creates a new class which will create a
 * deep copy of the input B object 'right'
 */
```

```
/** Destructor
  * Correctly deallocates all object resources
  */
```

22. A regular polygon is a shape with any number of sides, where all sides are the same length, and all angles are equal in degrees.

```
1  class RegularPolygon {
2      public:
3          RegularPolygon();
4          RegularPolygon(int numSides, int sideLength);
5          virtual ~RegularPolygon();
6          virtual int numberOfSides();
7          virtual int area();
8      private:
9          double interiorAngle;
10         int sideLength;
11 }
```

(a) [5 points] Write the class declaration for a subclass of RegularPolygon called Square. This subclass will have an attribute to keep track of its current area, to speed up the area() function, which it must override from the base class. The subclass must also define a default constructor, and a one parameter constructor which takes the side length as a parameter.

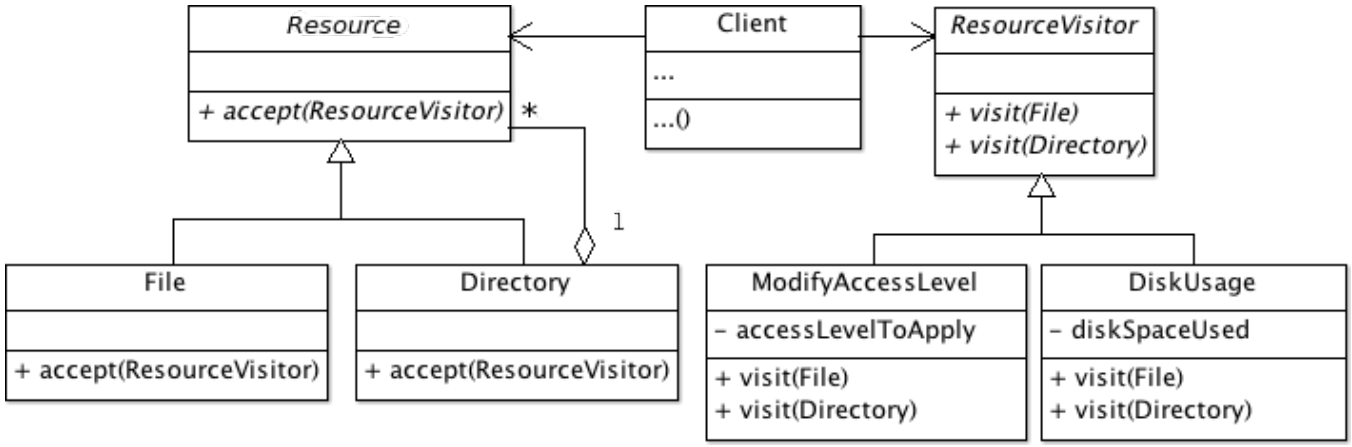
... Continued from previous page

(b) [5 points] Provide an implementation for the one parameter constructor, which calls the base class constructor as appropriate. The constructor must initialize all its attributes, and all the attributes in the base class.

(c) [5 points] Does the above example violate the Liskov Substution Principle? Give a basic description of the Liskov Substitution Principle and explain why there is, or is not, a violation.

23. [15 points] Compare and contrast the State and Strategy design patterns. Draw a UML diagram outlining each pattern, and briefly describe the context in which each are used. Include some additional points comparing the two patterns, and describe in what situation you would use each over the other.

24. Consider the class diagram below:



(a) [3 points] Identify the design patterns used in the following system and identify their types (Creational, Behavioural, or Structural).

(b) [4 points] Describe one advantage of having applied either of the two design patterns to this context. The advantage must be a specific advantage of using that pattern, rather than a general benefit of using a design pattern.

... Continued from previous page

(c) [8 points] Summarize what you believe the purpose of the system is, and how the design patterns are used to implement this. Use example(s) if you would like.