Polytechnic Tutoring Center

Final Exam Review - CS 2124 Spring 2026

Disclaimer: This mock exam is only for practice. It was made by tutors in the Polytechnic Tutoring Center and is not representative of the actual exam given by the CS Department.

1.

```
class A {
private:
	string aString;
protected:
	const string& get_string() const {
		return aString;
	}
public:
	A(const string& anotherString) : aString(anotherString) {}
};

class B : public A {
public:
	B(const string& otherString) : A(otherString) {}
	void display_string() const {
		cout << get_string() << endl;
	}
};

class C : public A {
public:
	C(const string& anotherotherString) : A(anotherotherString) {}
	void display_string() const {
		cout << aString << endl;
	}
};
int main() {
	A a("A");
	B b("B");
	C c("C");
	b.display_string();
	c.display_string();
}
```

What is the result of the above program?

a. Runtime error as a result of Class B's display_string method

b. Runtime error as a result of Class C's display_string method

c. The output: B
   C

d. Compilation error as a result of Class C's display_string method

e. Compilation error as a result of Class B's display_string method

2. What does the following code result in? And what key concept is used in this code?

```
class A {
private:
	string aString;
public:
	A(const string& anotherString) : aString(anotherString) {}
};

class B : public A {
public:
	B(const string& otherString) : A(otherString) {}
};


int main() {
	A* b = new B("B"); \\ line A
	B* a = new A("A"); \\ line B
}
```

a. Results in a compilation error at line A; polymorphism
b. Results in a compilation error at line B; polymorphism
c. Results in a compilation error at line A; abstraction
d. Results in a compilation error at line B; abstraction

3. **What is the result of the following code? Output on the line below…**

```
vector<int> vi = { 10, 2, 4, 1, 5, 3, 7, 9, 8, 6 };
list<int> li(3 + vi.begin(), vi.end());
for (int elem : li) {
	cout << elem << " ";
}
```

4. What will be the output of the following code?

```
class A {
private:
string aString;
protected:
	const string& get_string() const {
		return aString;
	}
public:
	A(const string& anotherString) : aString(anotherString) {}
	~A() { cout << "~A()\n"; }
};

class B : public A {
public:
	B(const string& otherString) : A(otherString) {}
	void display_string() const {
		cout << get_string() << endl;
	}
	~B() { cout << "~B()\n"; }
};


int main() {
	A* a = new B("B");
	B* b = new B("B2");
		delete a;
		delete b;
}
```

A: ~A() ~B()	B: ~A() ~B() ~A()	C: ~B() ~A() ~B()	D: ~B() ~A() ~B() ~A()	E: ~A() ~B() ~A() ~B()

5. Write a templated function that enables you to find a specified value within any container with an iterator. You can assume the container only stores objects of one type. Note this “specified value”, along with the beginning and end should be passed into the function. If you couldn’t find the value, return a reasonable value.

---

6. Write a recursive function that returns whether the string passed into the function is a palindrome. Note: you are allowed to use helper variables like the typical high and low frequently seen in data structures.

---

7. Given the following tree node data structure implementation:

```
struct TNode {
    int data = 0;
    TNode* left = nullptr;
    TNode* right = nullptr;
};
```

Every TNode object lives on the heap. Write a recursive function, given a root node, free up the dynamically allocated memory by destroying all the nodes connected to it.

---

8. Write an Iterator class that iterates over a singly linked list. Assume the iterator class you are writing is of the non-const type. Write the pre and post increment operators as well as the operator== and operator!= operators for the iterator. Assume the class has the following Node struct definition:

```
struct Node {
       Node* next;
       int data;
       Node(Node* next = nullptr, int data = 0) : next(next), data(data) {}
};
```

---

9. The following problem involves three classes: MapScreen, RadarMap and Coordinate. You are only responsible for implementing the RadarMap class.

MapScreen class

Note: *You are not responsible for implementing the MapScreen class. The MapScreen class has a private string representing the MapScreen's title*

- Has a constructor accepting the MapScreen's title
- Supports copy control
- Has a get_title() method that returns the MapScreen's title
- May have additional fields and methods

Again, note: you are not responsible for implementing the MapScreen class

RadarMap class

RadarMap is a *derived* class of the MapScreen class used to represent many different coordinates.

- The RadarMap class contains the following member variables: a private string representing the color of coordinates on the map screen
- a private bool indicating whether or not the map screen is currently being displayed with a default value of false
- a private dynamic array of Coordinate pointers
- All Coordinate objects in the array will be stored on the heap.
- The Coordinate objects are "owned by" the RadarMap instance, i.e., no one else has a pointer

to these Coordinate objects.

- The Coordinate objects in the array represent the coordinates installed on the RadarMap.
- In the constructor, initialize the array with a size of 100.
- The Coordinate class supports copy control and all necessary operators.
- Do not assume the existence of any other methods for the Coordinate class.

You do not have to implement the Coordinate class!

Note*: Assume the vector passed into the constructor always has a 100 Coordinate objects When you need the size of the array, use a 100. All 100 parts of the array are filled.*

What you have to do however are as follows:

Define the RadarMap class. You are only responsible for implementing the RadarMap class' functionality as follows:

1. Implement the General Constructor
2. Implement a constructor that accepts the RadarMap's title, color of coordinates, and display status and a vector of pointers which you will have to add to your array.

(remember you would have to add these coordinates on the heap to the dynamic array in your private member class) (The coordinate class has a copy constructor you can use while initialization)

3. Implement Copy Control
    a. Implement all of the copy control.

2. Implement an output operator. You choose the format, but the title, color of coordinates, and the coordinates contained in the RadarMap must all be neatly displayed.

3. Implement an Index Operator
   Implement an index operator that can access a Coordinate pointer in the RadarMap's coordinates array based on the value passed to an index parameter. The behavior of the index operator is as follows:

- In the case that the index parameter has a value greater than or equal to the number of Coordinates in the array, the function returns nullptr.
- Otherwise, the Coordinate pointer at the index passed as the value of the parameter is returned.
- The array of pointers cannot be modified using the index operator.
- Consider the statement below:

```
const Coordinate* npc = radarMap[35];
```

If 35 is a valid index (i.e., less than 100), npc will be assigned the address of the Coordinate at index 35 of the array. If the index is 100 or greater, the index operator returns nullptr.

4. Implement a find() method

- The find () method accepts a Coordinate object and returns true if the Coordinate is contained in the RadarMap on which its invoked and false otherwise.
- Implement the Boolean Operator
- When evaluated in a boolean context, the RadarMap object will indicate whether or not it is currently-displayed. You do not have to implement any behavior that displays or hides the RadarMap object.

The following code (rm is a RadarMap object) will compile and result in the correct output:

```
if (rm) { cout << "Currently displayed"; }
```

Note that based on your implementation, the following statement will result in a compilation error.

```
cout << rm + 1;
```

Bonus question:

Write a recursive function to convert a decimal to an Octadecimal number(Base 8). Print the number out. (function returns void then)