ANSWER KEY

Polytechnic Tutoring Center

Exam 1 Review - CS 2124, Fall 2021

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. Declare:

a.	An unchanging pointer to an int	**`int* const p;`**
b.	A pointer to an unchanging int	**`const int* p;`**
c.	An unchanging pointer to an unchanging int	**`const int* const p;`**

2. What will the following code result in?

```
int main(){
      int i = 7;
      const int* ip = &i;
      i += 1;
      cout << i << endl;
}
```

   a. 7
   b. **8**
   c. Runtime error
   d. Compilation error

3. What will the following code result in?

```
int main(){
      const int i = 7; //line A
      int* ip = &i; //line B
      cout << i << endl; //line C
}
```

   a. It will just print out 7
   b. It will have an error at line A
   c. **It will have an error at line B**
   d. It will have an error at line C
   e. The program will crash
   f. It is undefined

4. What will the following code result in?
```
int* foo(int x) {
      int y = 2*x;
      int* p = &y;
      return p;
}

int main(){
      int z = 29;
      cout << *foo(z);
}
```

   a. Compilation error
   b. Segmentation fault error
   c. Runtime error
   d. The program will crash
   e. **It is undefined**

5. Given the struct defined below, what would be the result of:
```
struct CovidStatus {
      int infected;
      int victims;
      int recovered;
protected:
      CovidStatus();
};

int main() {
      CovidStatus covid19;
      cout << covid19.infected << endl;
}
```

   a. Runtime error
   b. Compilation error
   c. It is undefined
   d. None of the above

6. Given that there is a class of `Thing` objects that have an attribute called `item`, and `tp` is a pointer to a `Thing`, what is the meaning of the following?

```
tp->item
```

   a. `*(tp.item)`
   b. `(tp.item)*`

c. `*tp(.item)`
d. **`(*tp).item`**

7. Given the struct definition below, write a function that iterates over a vector of `Account` objects and finds the `Account` object with a specified `acc_id`. The function should pass in the vector of `Account` objects as well as the `acc_id`, and then return the index of the matching `Account`.

```
struct Account {
      int acc_id;
      int balance;
      Account(int id, int amnt = 0) : acc_id(id), balance(amnt) {}
}

size_t find_acc(const vector<Account>& v, int id) {
      for(size_t i = 0; i < v.size(); i++) {
            if (v[i].acc_id == id) {
                  return i;
            }
      }
      return v.size();
}
```

8. Using the struct definition in question 7 for `Account`. Create a function that reads through a file and uses the contents of the file to modify an array that holds a maximum capacity of say 10 pointers to `Account` objects. The contents of the file would be similar to that seen below:

```
ID: 89203482
Balance: 124

ID: 09859403
Balance: 877

ID: 71934098
Balance: 1080
```

The output from the function, when called on this particular file, would yield an array holding three pointers to `Account` objects each with the `acc_id` and `balance` attributes as seen above. Note that the function should not return the array itself, rather it should pass in the array and add the pointers to it.

```
void read_accounts(ifstream& ifs, Account** accnts) {
      int id, bal;
      string blank1, blank2;
      size_t i = 0;
      while (ifs >> blank1 >> id >> blank2 >> bal & i < 10) {
            accnts[i] = new Account(id, bal);
            i += 1;
      }
}
```

9. Write a function that then iterates over this array of `Account` pointers and frees up all the memory within the array, including the array itself.

```
void clear_accounts(Account** accnts) {
      size_t i = 0;
      while ( i < 10 ) {
            delete accnts[i];
            accnts[i] = nullptr;
            i += 1;
      }
      delete[] accnts;
}
```

10. Write a function that increments all elements of a passed in vector of ints. You must use a ranged-based for loop to increment the elements.

```
void incrementElements(vector<int>& v){
      for(int& x:v)
            x++;
}
```

11. Write a class Employee to model the employees in a company PolyCo.
- In PolyCo, each employee has a name, can have only one boss and zero or more sub-employees.
- The CEO of PolyCo doesn't have a boss, of course, but every other employee does.
- When employees are created, there are the following two possibilities:
  - They can be told who their boss is right away.
  - They can be created without a boss.

ANSWER KEY

- Employees should be able to be added and removed from the list of sub-employees at any time in the future. Provide support for this. **Think about the fact that employees in the team need to know who their boss is, and the boss needs to know who's in his team.**
- Write a display function that will give the output as shown.
- **Important: Only write the class Employee!**
- **Assume that there will be no duplicate employees added to a boss employee.**
- **Since employees can have only one boss, if we hire someone who already has a boss, we should not let this happen and report a failure.**
- Here's a sample main and the sample output it produces, make sure your code, if ran, will have the same output as seen here.

```
int main()
{
      Employee sterling("Professor Sterling");
      Employee yan("Yan", &sterling);
      Employee jeremy("Jeremy");
      Employee mike("Mike");
      yan.addToTeam(jeremy);
      yan.addToTeam(mike);
      cout << yan << endl;
      cout << sterling << endl;
}
```

Output:

```
Name: Yan
Boss: Professor Sterling
Team...
      Jeremy
      Mike

Name: Professor Sterling
Boss: I am the boss.
Team...
      Yan
```

```
#include<iostream>
#include<string>
#include<vector>
using namespace std;

class Employee
{
```

ANSWER KEY

```
friend ostream& operator<<(ostream& os, const Employee& emp) {
      cout << "Name: " << emp.name << endl;
      cout << "Boss: ";
      if (emp.boss != nullptr)
      {
            cout << emp.boss->name << endl;
      }
      else
      {
            cout << "I am the boss." << endl;
      }
      cout << "Team..." << endl;
      for (size_t i = 0; i < emp.myTeam.size(); i++)
      {
            cout << "\t" << emp.myTeam[i]->name << endl;
      }
      return os;

}
private:
      string name;
      Employee* boss;
      vector<Employee*> myTeam;
public:
      //Constructor
      Employee(const string& name, Employee* boss = nullptr) :
      name(name),boss(boss)
      {
            if(boss != nullptr){
                  boss->myTeam.push_back(this);
            }
      }

      bool addToTeam(Employee& newGuy)
      {
            if (newGuy.boss != nullptr)
            {
                  return false;
            }
            newGuy.boss = this;
            myTeam.push_back(&newGuy);
            return true;
      }


      bool removeFromTeam(Employee& emp)
      {
            for (size_t i = 0; i < myTeam.size(); i++)
            {
                  if (myTeam[i] == &emp)
                  {
                        emp.boss = nullptr;
                        myTeam[i] = myTeam[myTeam.size()-1];
                        myTeam.pop_back();
                        return true;
                  }
            }
```

```
                return false;
        }
};
```