```
int* p;
int* q = p;
p = new int;
*p = 55;
std::cout << *q << std::endl;
```

Solution: This code contains a dereference of an uninitialized pointer. This may cause a segmentation fault at runtime, or unexpected output. Add q = p before the cout statement to fix the problem, change *q to *p or put a value in q before new p.

```
std::vector<std::string> > pets;
pets.push_back("cat");
pets.push_back("dog");
pets.push_back("elephant");

std::cout << pets[1] << " " << pets[2] << " " << pets[3] << std::endl;
```

Solution: An attempt was made to reference a vector element that was not allocated. The solution is to reduce each index by 1. There was also extra > on the first line, removing the extra > was another solution.

```
std::cout << pets[0] << " " << pets[1] << " " << pets[2] << std::endl;
```

```
std::vector<std::string>& Vectorfy(const std::string& s) {
  std::vector<std::string> v;
  v.push_back(s);
  return v;
}
```

Solution: The function is returning a reference to a local variable. Return a copy.

```
std::vector<std::string> Vectorfy(const std::string& s)
```

Solution:

```
bool WordInVector(const std::vector<std::string>& vec, const std::string& word, unsigned int& position,
                  unsigned int start_position){
    for(unsigned int i=start_position; i<vec.size(); i++){
        if (vec[i] == word){
            position = i;
            return true;
        }
    }
    return false;
}
```

Solution:

```
std::vector<int> count_phrase(const std::vector<std::string>& words, const std::string& phrase){
  std::vector<int> ret(words.size(),0);

  //Check each word
  for(unsigned int i=0; i<words.size(); i++){
    //Go letter by letter for starting position
    for(unsigned int j=0; j<words[i].size(); j++){
      unsigned int k;
      //Check if the substring is found starting at words[i][j+k]
      for(k=0; k<phrase.size() && j+k < words[i].size(); k++){
        if(words[i][j+k] != phrase[k]){
          break;
        }
      }

      //Found the whole phrase
      if(k==phrase.size()){
          ret[i]++;
      }
    }
  }

  return ret;
}
```