Hayden Fuller 662028619 Section 5 TA: Charles Mentors: Hanson & Theodore & Ali

sizeof() gives memory, the means the space a pointer takes up if given
Int = float will truncate, does not work in vectors

```
#include… iostream, cmath, cstdlib (exit),  string,
vector, fstream, algorithm
bool isThisTrue(int same, double &change,
std::string &large) { return true; }

int main(int argc, char* argv[]) {
  std::cout << “enter something” << std::endl;
  int something1, something2;
  std::cin >> something1 >> something2;
  std::cerr << “wrong something” << std::endl;
  int a[8];
  a[2] = 16;
  std::string stars(12,’*’);
  stars2 = stars.c_str();
  char h[] = “HW!”; //OR {‘H’, ‘W’, ‘!’, ‘\0’};
  std::string h2(h);;
  std::vector<int> v(10, 5); //5,5,5,5,5,5,5,5,5,5
  std::vector<itn> c(v); //copy
  std::sort(c.begin(), c.end(), optional); // no ()
  for(int i = 0; i < 2; i++) { while(1) { break; } }
  std::ifstream file_in(“input.txt”);
  std::ofstream file_out(“output.txt”)
  if(!file_in.good()) { std::cerr << error; exit(1); }
  file_in >> s >> s1;    file_out << h2 << “hello”;
  Int x; while(file_in >> x) { v.push_back(x) }
  const int n = 10; int *p = &x;  int a[n];
  for(p=a; p<a+10; p++){ *p=sqrt(p-a);}
  int *a = new int[n]; for(int *p = a; p<a+n; p++){
  int** a = new int*[r];  for(int i=0;
  i<r;i++){a[i]=new int[c]; for(int
  j=0;j<c;j++){a[i][j]=int(i+1)/int (j+1);}}
  int readInt; int* intArray=new int[max];
  while(input>>readInt){*(intArray +
  *numElements) = readInt; *numElements += 1; }
  str.substr(start, length);   //npos = end of string
  str.find(“world”);          //returns iterator of place
  //students.push_back(Student(name, age));
  Student stu(“name”, 19);
  std::cout << stu << std::end;
  return 0; }
```

In student.h

```
#ifndef __student_h_
#define __student_h_
#include …
class student {
public:
  Student();
  Student(std::string aName, int aAge);
  std::string getName() const;
  int getAge() const;
  void setName(std::string aName);
  void setAge(int aAge);
  bool sameName(const Student& s2) const;
  void print() const;
private:
  std::string name; int age;
}
bool operator< (const Student& s1, const Student&
s2);
std::ostream& operator<< (std::ostream& ostr, const
Student& s);
#endif
```

In student.cpp

```
#include…   #include “student.h”
Student::Student() {  name = “No-name”; age = 0; }
Student::Student(std::string aName, int aAge) {
name = aName; age = aAge;}
Std::string Student::getName() const
{ return name; }
int Student::getAge() const { return age; }
void Student::setName(std::string aName)
{ name = aName; }
void Student::setAge(int aAge) { age = aAge; }
bool Student::sameName(const Student& s2) const {
//check }
bool operator< (const Student& s1, const Student&
s2) { /* sort */ return true; }
std::ostream& operator<< (std::ostream& ostr, const
Student& s) { ostr << s.getName() << “ - “
s.getAge() <<std::endl;   return ostr; }
```

```
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;
}
```