CSCI-1200 Data Structures — Spring 2023
Test 1 — Thursday, February 2nd 6-7:50pm

Ryan So	sor@rpi.edu lab section: 6
room: Darrin 337 zone: BROWN row: 4 seat: 7	6-7:50pm

Write the name of one of your undergraduate mentors.

Terry

What time does your lab section end at?

4:00 pm

- This exam has 4 problems worth a total of 100 points (including the cover sheet).
- This packet contains 8 pages of problems numbered 1-8. Please count the pages of your exam and raise your hand if you are missing a page.
- The packet contains 2 blank pages. If you use a blank page to solve a problem, make a note in the original box and clearly label which problem you are solving on the blank page.
- This test is closed-book and closed-notes except for the .pdf notes you (optionally) uploaded to Submitty by last night. These notes are the last 2 pages of your exam packet.
- **DO NOT REMOVE THE STAPLE OR SEPARATE THE PAGES OF YOUR EXAM. DOING SO WILL RESULT IN A -10 POINT PENALTY!**
- You may have pencils, eraser, pen, tissues, water, and your RPI ID card on your desk. Place everything else on the floor under your chair. Electronic equipment, including computers, cell phones, calculators, music players, smart watches, cameras, etc. is not permitted and must be turned "off" (not just vibrate).
- Raise your hand if you need to ask a proctor something that is not related to one of the questions on the test. We will **not** be able to answer if it about one of the test problems.
- Please state clearly any assumptions that you made in interpreting a question. Unless otherwise stated you may use any technique that we have discussed in lecture, lab, or on the homework.
- Please write neatly. If we can't read your solution, we can't give you full credit for your work.
- You do not need to write `#include` statements for STL libraries. Writing `std::` is optional.

1 Short Answer Round [/ 24]

For each of the following statements, write if it is true or false, and then write 1-2 *complete* sentences explaining why. Most of these statements are false.

Reminder: Write complete sentences!

1.1 Big Parameters [/3]

True or False Variables larger than 8 bytes should always be passed in by constant reference.

True. Variable that aren't passed in by const ref, the computer will make a copy. Creating a copy uses extra memory.

1.2 Default private [/3]

True or False If we do not use the public or private keyword in a class definition, then all members of the class will be treated as private.

False, They would be treated as public rather than private. By specify some variables are private, only then do they become private.

1.3 Member Output Operator [/3]

True or False We should write the output stream operator<< for our own class Foo as a member function of Foo, so we can have access to private member variables of Foo when printing.

False, It isn't necessary to write your own stream operator to output private members.

1.4 Returning Constant Reference [/3]

True or False When returning a string, we should never use a const& string return type.

False. f

1.5 Sorting Tie Breakers [/3]

True or False When writing a custom sort function to pass into `std::sort`, if there is a tie the function must return `return false;`.

True, If there is a tie, then it isn't true. Therefore it's false.

1.6 Default Constructor? [/3]

True or False If a class Foo only has one constructor in the class declaration, `Foo(int x=5)`, we cannot declare a instance like `Foo f;` because there is no default constructor.

False. Foo(int x=5) is already a default constructor.

1.7 Cleaning Up Memory [/3]

True or False There are no memory leaks in the following code:

```
int* p = new int;
int* q = p;
delete p;
```

False. When p is deleted, q isn't pointing to anything anymore and the new int stays in the heap unused.

1.8 Array Direction [/3]

True or False Arrays have their index 0 at the highest (largest) memory address and for an `int a[n]` where `n` is a positive number, the valid range of addresses is `(a-n, a]`.

False, arrays store their first index at the lowest memory address.

2 Clown Class [/ 32]

In this problem you will write a Clown class to represent students studying at a clown school to learn skills. Each clown has a name and keeps track of the skills they learn. You can assume no two clowns will have the same name.

Skills are taught through the `learn_skills` function. A clown cannot learn a skill they already know. Sorting by `order_by_skill` function should sort by the number of skills, resolving any ties by name. The following example illustrates how the class should work:

```
Clown a("Applejack"), b("Fluttershy"), c("Rarity");
assert(a.get_name() == "Applejack");
assert(a.skill_count() == 0);

bool success = b.learn_skill("Juggling");
assert(success && b.skill_count() == 1);

std::vector<std::string> skills({"Juggling", "Tightrope"});
int newly_learned = b.learn_skills(skills);
assert(newly_learned == 1 && b.skill_count() == 2);

newly_learned = c.learn_skills(skills);
assert(newly_learned == 2);

std::vector<Clown> clowns;
clowns.push_back(c);
clowns.push_back(b);
clowns.push_back(a);

std::sort(clowns.begin(), clowns.end(), order_by_skill);

for(unsigned int i = 0 ; i < clowns.size() ; i++){
  clowns[i].print(std::cout);
}
```

Which produces the output:

```
Applejack knows 0 skills.
Fluttershy learned 1 skills.
Rarity learned 2 skills.

Summary:
Fluttershy has learned Juggling, Tightrope
Rarity has learned Juggling, Tightrope
Applejack has learned no skills
```

2.1 Clown class declaration (Clown.h file) [/12]

Write the declaration for the Clown class and any non-member functions. We recommend that for functions that will only have a single short line of implementation, you place the implementation in this file. Remember that longer function implementations should be put in the .cpp file in the next question.

```
class Clown {
      public:
      Clown( string name);
       string get_name() const;
       Vector<string> getskills() const;

      bool learn_skills( string skill);
      int  skill_count() const;
      Void print (std::cout);


      Private:
      string name;
      vector<string> skills;
};
bool order_by_skills(const Clown &clown1, const Clown &clown2);
```

sample solution: 12 line(s) of code

2.2 Non-member functions (Clown.cpp) [/6]

Write the implementation of any non-member functions for the Clown class. You do not need to put any #include statements in this part.

```
bool order_by_skills(const Clown& clown1, const Clown& clown2){
      if (clown1.skill_count() != clown2.skill_count()){
         return (clown1.skill_count() > clown2.skill_count());
      } else{
         return (clown1.get_name()< clown2.get_name());
      }
```

sample solution: 8 line(s) of code

2.3 Clown implementation (Clown.cpp) [/14]

Write the implementation of the Clown class's member functions.

```
Clown::Clown( string aname){
      name = aname;
      skills = ();
}

String  Clown::get_name()const {
        return name;
}

Vector<String> Clown::getskills() const {
       return skills;
}

bool  Clown::learn_skills(string skill){
      int skill_size = skills.size();
      for (int i=0; i< skill_size; i++){
          if (skills[i] == skill){
             return false;
          } else if((skills[i] != skill) && (i+1 == skill_size)){
             skills.push_back(skill);
             return true;
          }
    }
}
int  Clown::skill_count() const {
     int skill_size = skills.size();
     return skill_size;
}
void Clown::print(std::cout){ cout<<name<< " has learned ";
      if(skills.size()==0){ cout<< "no skills"<<std::endl; }
      else { for(int i=0; i< skills.size(); i++){ if(i==0){ cout<<skills[i]<<'
      }else{cout<< ','<< skills[i]}} cout<<endl}}
```

sample solution: 29 line(s) of code

3 Acronym Finder [/ 24]

In this question, we define the *acronym* of a phrase to be the string that is formed when taking the uppercase first letters of each word (as separated by spaces) in that phrase. For example, "Data Structures", "Denial of Service", and "D S" are phrases that have the acronym "DS". The phrase "DataStructures" has the acronym "D" instead of "DS", because the lack of spaces means it is treated as one word. "Data Structures and Algorithms" has the acronym "DSA" because we skip "and" since it does not start with a capital letter.

Fill in the function prototype for `isAcronym()`, which takes in two strings, one being the phrase and other being the acronym, and returns a boolean indicating whether the phrase's acronym is the one that's supplied. Also fill in the function implementation. You cannot declare any additional variables or loops, you must make use of the arguments, the two `int`s we declare, and the single `while` loop.

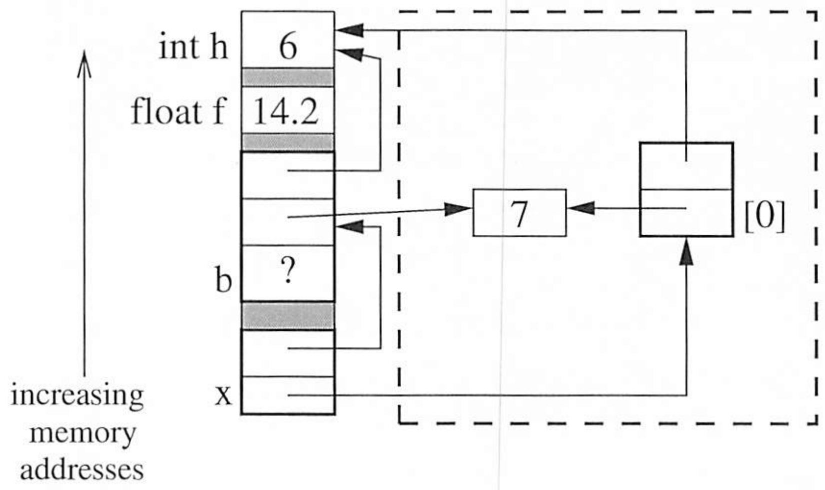
You can assume the phrase only has letters and spaces, and that the acronym only has capital letters. Here are a few usage cases:

```
isAcronym("Data Structures", "DS")); //true
isAcronym("Data Structures and Algorithms", "DSA")); //true
isAcronym("DataStructures", "DS"); //false
```

```
bool isAcronym( String phrase , String acr ) {

  unsigned int phrase_pos = 0;
  unsigned int acronym_pos = 0;

  while( phrase_pos < phrase.length() || acronym_pos < acr.length() ){
    if ( phrase[0] != acr[0] ){
       return false;
    } else if ( phrase[phrase_pos] == ' ' ){
       if ( phrase [phrase_pos +1] != arc[acronym_pos +1] ){
          return false,
       }
    } else {
       phrase_pos +1;
    }
                                          sample solution: 10 line(s) of code
  } // End of for loop

  return ( phrase_pos == phrase.length())&& (acronym_pos+1 == acr.length());
}
```

4 Memory Drawing [/ 17]

Write code to produce the memory diagram shown on this page. Some types have been left out. Thicker bold boxes indicate arrays.



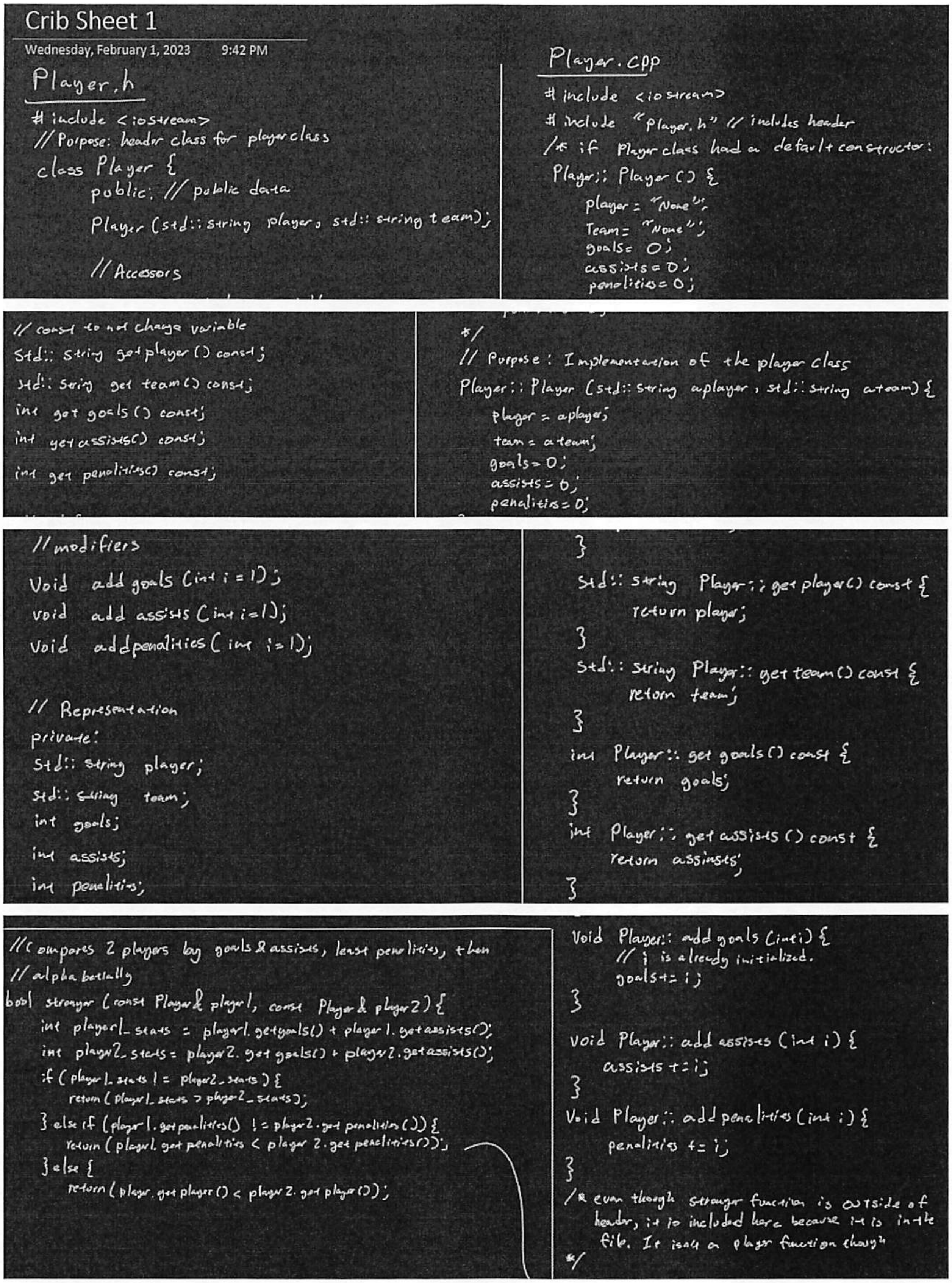
```
int h=6; *b[3], **x [2];
float f=14.2;
x[0] = new int [2];
x[1] = *b[1];
x[0][0] = new int;
*x[0][0]=7;
x[1][1] = x[0][0];
b[2]=h;
x[0][1]=h;
```

After the above code has been run, is it possible to clean up all dynamically allocated memory? If it is not, write 1-2 sentences explaining why. Either way, write code that will clean up as much of the heap memory used as possible.

```
delete [] x[0];
delete [] x;
```

b[0] is a pointer, but never points to anything. Although all memory in the heap is deleted, there is memory not being used.

Ryan So sor@rpi.edu



Crib Sheet 1

Wednesday, February 1, 2023 9:42 PM

Player.h

```
# include <iostream>
// Purpose: header class for player class
class Player {
        public: // public data
        Player (std::string player, std::string team);

        // Accesors
// const to not change variable
std:: string getplayer () const;
std:: string get team () const;
int get goals () const;
int get assists() const;
int get penalities() const;

 //modifiers
 Void add goals (int i = 1);
 void add assists (int i=1);
 Void addpenalities ( int i=1);

 // Representation
 private:
 Std:: string player;
 std:: string team;
 int goals;
 int assists;
 int penalities;
```

```
//Compares 2 players by goals & assists, least penalities, then
// alphabetically
bool stronger (const Player& player1, const Player& player2) {
    int player1_stats = player1.getgoals() + player1.getassists();
    int player2_stats = player2.get goals() + player2.getassists();
    if ( player1_stats != player2_stats ) {
        return ( player1_stats > player2_stats );
    } else if (player1.getpenalities() != player2.get penalities()) {
        return (player1.get penalities < player 2.get penalities());
    }else {
        return ( player.get player () < player 2. get player());
```

Player.cpp

```
# include <iostream>
# include "player.h" // includes header
/* if Player class had a default constructor:
Player:: Player () {
    player = "None";
    Team = "None";
    goals = 0;
    assists = 0;
    penalities = 0;
*/
// Purpose: Implementation of the player class
Player:: Player (std:: string aplayer, std:: string ateam) {
    player = aplayer;
    team = ateam;
    goals = 0;
    assists = 0;
    penalities = 0;
  }
  Std:: string Player:: get player() const {
        return player;
  }
  Std:: string Player:: get team() const {
        return team;
  }
 int Player:: get goals () const {
      return goals;
 }
 int Player:: get assists () const {
      return assinses;
 }
 Void Player:: add goals (int i) {
      // i is already initialized.
      goals+= i;
 }
 void Player:: add assists (int i) {
    assists += i;
 }
 Void Player:: add penalities (int i) {
     penalities += i;
 }
 /* even though stronger function is outside of
   header, it is included here because it is in the
   file. It isn't a player function though
 */
```

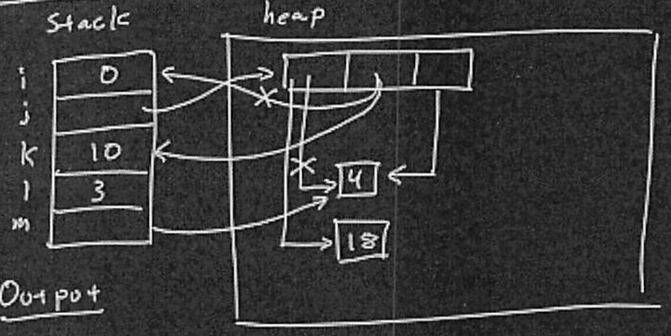
Ryan So sor@rpi.e

Memory Diagram

Code:

```
int i, **j, k, l, *m;
i=0;
j= new int *[3];
j[0] = new int;
j[1] = &i;
m= *(j+1);
j[1]= &k
k=10;
*(j[0])= 5;
j[2] = j[0];
*(j[0]) = 18;
*m=4;
l=3
```

Diagram



Output

```
i: 0
j[0]: 18
j[1]: 10
j[2]: 4
k: 10
l: 3
m: 4
```

Strings   # include <ctypes>

```
string.isalpha()  string.isdigit()
//returnse bool whether string is letter/digit
String.islower()   string.isuppr()
 // returns bool whether string is lower/upper
String.tolower()   string.toupper()
```

Vectors

```
#include <vector>
std:: vector <int > a(10); // vector size 10 of ints
a.back(4); //a[-1] = 4
a.pop.back(); //vector size is now 9
a.front(); // a[0]
a[20]; // memory not given to you, seg fault
```

File reading

```
# include <fstream>
std::ifstream in_str("file.txt"); // reads files
std::ofstream out_str("file.txt"); // writes files
if (!in_str.good()){ // checking if text file exist
    std::cerr << "Can't Open" << std::endl;
    exit(1);

in_str >>x;
Out_str << y << std::endl;
in_str.close();
out_str.close();


Sizeof(x) // tells you how many bytes it holds.
          // divide by variable type to get number of elements
```

Consts & &

```
& //reference. use in function so time is less
......const() // won't modify variables.
              // no affect on run time.
const .....  // passes a reference and no
             // changes.
```

Errors

std has no member → missing an include, not found in library
redefinition of "class Class" → add guards to class.h file
"reference to local variable" → remove & to variable
"control reaches end of non-void function" → force return a
Stack Memory
Un initialized Read → didn't declare "new"
unaddressable Access → accessing past array size
Invalid heap argument & LEAK → didn't use delete