CSC 340.03+.04+.05 MIDTERM EXAM 02 FALL 2020 TA

MIDTERM EXAM INSTRUCTIONS

1. Midterm Exam: **25 points w/ 0 E.C. points**
2. Due Date & Time: **11-22-2020 at 11:55 PM**

WHAT TO SUBMIT

1. Take-home Exam Report, 1 PDF

HOW TO SUBMIT AND THE RULES TO FOLLOW

- Submit via iLearn, the Assignment and Exam Submission section
- Please follow the exam instructions
- Please follow the Course Policy on Student Conduct and Academic Honesty

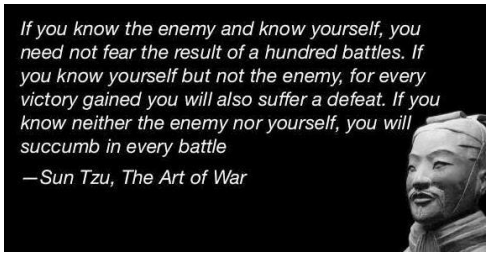
A: 90-100% **B**: 80-89% **C**: 70-79% **D**: 60-69% **F**: 0-60% The course grader provides feedback to your assignments on iLearn.

PERFORMANCE TRACKER

ASMT	GRADE	YOUR GRADE
ZOOM	05	
01	15	
02	100	
03	100	
MIDTERM 01	25	
04-PREPARATION	25	
04	75	
MIDTERM 02	25	
TOTAL	370	

ABOUT

The goal of this take-home exam is for us to **know what we do not know**.



We are taking this exam as seriously as how we take an actual exam in class. Please,

1. Follow all the rules and the guidelines listed at the top of page 1 and page 2
2. Read each question carefully before answering

We will go through the answers to all the exam questions together in class.

STEP A – Take the Exam, **10 points**

1. Allocate 50 quiet minutes to take the exam on page 2 to the last page.
2. Record the date and time when you start.
3. Stop right at minute 50. Record the date and time when you stop the exam.

STEP B – Correct Your Answers, **10 points**

1. Review the related course materials and write code when necessary to find a correct answer for each question. We should be able to find all the answers using the packages, the in-class discussions, our assignments, and the other course materials.
2. At the end of each of your oringal answers, type in *italic* text and:
   - Give your orginal answer a score.
   - List all the mistakes then explain why, you think, you made the mistakes. Add the correct answer you found. Document how you found the correct answer. Document where you found the materials which support the answer.
   - If you did not make any mistakes, please document how you verified that your answer was accurate. Document how you found the correct answer. Document where you found the materials which support the answer. Outline how you could have done better.
   - Record your total score out of 100 points for all the orginal answers.

STEP C – Reflect and Retake the Exam, **5 points**

1. **Problem Solving**: Reflect if you managed the exam time efficiently and if you strategized your test-taking successfully.
2. Repeat steps A to C again if necessary. Please keep appending new contents as directed in Step B.2.

3. Think if the same topics will be tested again in our final exam, what questions we may get.

It is a good idea to do every step of this assignment thoroughly. We are creating a set of materials which we will use to review for the final exam. And this is also the best way to prepare ourselves to succeed in the second half of the semester. Thank you.

1. Section, Date and Time: ##-####-#### at ##:## PM — Full Name in **CAPITAL LETTERS** | **SFSU ID** — TIC, TAC, and TOE. Due ##-

KAYVAUN KHOSHKHOU	920357344

2. Midterm Exam (2 exams, 0 dropped): 100 points
3. To prepare for this exam, please review all the related materials including WEEK 01-08 packages, slides, mock-up exam(s), reading assignments, in-class practices, sample programs posted in the File Manager, and assignments.
4. You do not need to print this exam. No paper. No handwriting. No scanning. Please type up all your answers in the answer space available in the exam. The provided exam will be in Microsoft Word format. Please submit a single PDF via iLearn.
5. All the rules of an actual exam apply to this exam such as: closed books, closed notes, and no communication with anyone except the course instructor. The course instructor will be available on Zoom or email during the exam time: zoom.ducta.net
6. Please ask all your questions, if any, during the review sessions. Thank you.

HONOR CODE:

- Please follow the CS Department's policies: https://cs.sfsu.edu/student-policies
- Please follow the course's policies: http://csc340.ducta.net/00-README-StudentConduct_AcademicHonesty.pdf

---

SECTION: TAC
11/22/20
EXAM START: 4:10 PM

PART A – 50 Points

A.1 - 10 pts

Please explain in detail Scott Meyers's point: Use weak_ptr for shared_ptr like pointers that can dangle.

Using weak_ptr for shared_ptr like dangling pointers is a bit ironic. For one thing, weak_ptr is created from shared_ptr. Weak_ptr also points to the same location as shared_ptr initializes them at. We can use weak_ptr for when we are working with observer lists, caching, and preventing shared_ptr infinite loops. It's important to keep their lifespans relatively similar because it is important in terms of memory management. You don't want your weak_ptr to outlive its partnered share_ptr because this can give you errors in the long run. Weak_ptr also does not affect the reference count of whatever object they point to. This is important because the cache will require a pointer to point to the objects. Those pointers need to be able to detect when they are dangling.

Answer matches that within the packages. 10/10

A.2 - 10 pts

How are Smart Pointer functions **move()**, **reset()**, and **release()** different from each other? Please also explain in detail which function is most dangerous and why?

Move() – the move() smart pointer is used to move object ownership instead of making a copy of it.
Reset() – the reset() smart pointer is used to set an object to null similarly to that if it was constructed by default.
Release() – the release() smart pointer is used to return ownership of a stored pointer and sets the pointer back to null.

These smart pointers are different in these main fundamentals, however the most "dangerous" one to consider would probably have to be the move() smart pointer. I believe this because it is a permanent instruction, once the object ownership and control has been moved it is very difficult to reverse.

Answer matches that within the packages. 10/10

- 16 Points

```
...
class Name {
public:
     Name() {}

    Name(string name) {
          this->name = name;
       }

     ~Name() {
       cout << this->name << ": Destructor called." << endl;
       }

   string getName() const {
          return this->name;
       }
private:
    string name{ "N/A" };
};

void passByMove(const unique_ptr<Name>
uPtr_M) {          cout << "@uPtr_M: " <<
uPtr_M << endl;
 cout << "getName(): " << uPtr_M->getName() << endl;
}

void passByRef(const unique_ptr<Name>&
uPtr_R) {          cout << "@uPtr_R: " <<
uPtr_R << endl;
 cout << "getName(): " << uPtr_R->getName() << endl;
}

void passByShare(const shared_ptr<Name>
sPtr_S) {          cout << "@sPtr_S: " << sPtr_S
<< endl;           cout << "getName(): " <<
sPtr_S->getName() << endl;
 cout << "use_count(): " << sPtr_S.use_count() << endl;
}

Name* passByValue(const unique_ptr<Name>
uPtr_V) {          cout << "@uPtr_V: " << uPtr_V
<< endl;           cout << "getName(): " <<
uPtr_V->getName() << endl;         return
uPtr_V.get();
}

int main() {

   cout << passByValue(make_unique<Name>("Goofy"))
                                                  << endl;

 unique_ptr<Name> uPtr{ make_unique<Name>("Mickey") };
```

```
        passByRef(uPtr);
cout << "name_uPtr: " << uPtr <<
endl;

  passByMove(move(uPtr));
 cout << "name_uPtr: " << uPtr << endl;

        uPtr = make_unique<Name>("Minnie");    shared_ptr<Name> sPtr{ uPtr.release() };  passByShare(sPtr);

 cout << "END of Program" << endl;
    return 0;
}
```

How many lines does this program output? ____16_____

Please give the **output** of the program. *Use **@A, @B, @C, @D**, and **nullptr** to represent memory addresses.*

01	@uPtr_V: @A
02	GetName(): Goofy
03	Goofy: Destructor called.
04	@A
05	@uPtr_R: @B
06	GetName(): Mickey
07	Name_uPtr: @B
08	@uPtr_M: @B
09	GetName(): Mickey
10	Mickey: Destructor called.
11	Name_uPtr: nullptr
12	@sPtr_S: @A
13	GetName(): Minnie
14	use_count(): 2
15	END of Program
16	Minnie: Destructor called.

Correct answer in lines (16), and correct output of program. 16/16

A.4 - 4 pts
Please explain in detail how to manually destroy an existing Smart Pointer control block.

In order to destroy and existeing smart pointer control block manually we must use the delete operator. The object will be destroyed by using delete-expression or a custom delete that is made for shared_ptr during construction of said code. Shared_ptrs have a feature where they are designed to delete the object they manage by default once it is obsolete. This will happen when no shared_ptr points to the object. So in order to manually destroy an existing smart pointer control block we can point the smart pointer to something else, thus deleted its previous pointed location, or we can use manual delete expressions in our program. For example like ; "delete[] ptr;" to delete multiple blocks or just "delete ptr;" to delete one block.

Answer matches that within the packages. 4/4

A.5 - 10 Points

```
…
int funcB(int);

int funcA(int n)
{           if (n
<= 1)

 return 217;         else
         return n + funcB(n - 2);
}

int funcB(int n)
{           if (n
<= 2) {
           return 3;
            } else {
            if (n >
4) {
                                  return
n * funcA(n - 5);               } else
{
                return n - funcB(n - 1);
            }
      }
}

int main() {
    cout << funcA(13);
     return 0;
}
```

What is the output of this program? **Please show our work.**

Output: 123

Work: were given cout << funcA(13);
n is not less than or equal 1 so

n+funcB(n-2) = 13 + funcB(11)
then we have
n * funcA(n-5) = 11 * funcA(6)
n + funcB(n-2) = 6 + funcB(4)
n – funcB(n-1) = 3 – funcB(2)
funcB(2) n is equal to 2 so it will return "3".
funcB(2) = 3;
funcB(3) = 3 – funcB(2)
3-3 = 0
funcB(4) = 4 – funcB(3)
4-0=4
funcA(6) = 6 + funcB(4)
11*10 = 110
funcA(13) = 13 + funcB(11)
13+110 = 123
Output: 123

Correct output given after running code through IDE to check. Explanation was done correctly showing all work. 10/10

PART B – 50 Points **B.1**

- 20 Points

```
... class
Name {
public:
 Name() {} private:
 string name{ "CS" };
};
shared_ptr<Name> func() {
 unique_ptr<Name> obj{ make_unique<Name>() };
 // #1 Insert Code
}
int main() {
 // #2 Insert Code
}
```

[#1 Insert Code]: **Write code** to keep the object which **obj** owns alive outside of the scope of function **func**. *Hint: The code should also support our task in #2 Insert Code.*

```
shared_ptr<exam> func(){
	unique_ptr<exam> obj{
		make_unique<exam>()
	};
	return obj;
}
```

[#2 Insert Code]: **Write code** to test if the object owned by **obj** is alive in the scope of function main. If it is, please output its address. If not, please output "Object destroyed."

```
int main(){
	shared_ptr<exam> obj = func();
```

```
        if (obj){
                cout << obj << "\n";
}
else{
        cout << "Object has been destroyed." << "\n";
}
```

Code checks out, after reading over the code I believe it fulfills the requirement. 20/20

B.2 – The Problem, 5 Points

Similar to what we did in ASMT 3, we are to carry out the three following steps. We use

1. **Copy constructor** to create a new bag using a bag passed in. However, **please note** that the order in which this constructor stores the items is the **reverse** of the original order (in which the items appear in the passed-in bag).
2. Function **addVector,** which takes a vector as the only parameter, to add the entries from the vector to the new bag.
3. Function **removeLastThree**, which takes an item as the only parameter, to remove the last three occurrences of the item in the bag. *It is OK to make assumptions. Please state our assumptions, if any.*

Please explain in detail **how each step works** and **what** the new bag contains after each of the steps is executed. -
Please use linked Nodes diagrams in our answer.
- Please use the data below in our explanation.

1. Passed-in bag: 'e', 'l', 'e', 'c', 't', 'r', 'i', 'c', 'a', 'l'
2. Passed-in vector: 'e', 'n', 'g', 'i', 'n', 'e', 'e', 'r'
3. Passed-in item: 'e'

Copy constructor (reverse order)

The bag is being reversed, so everything listed from left to right will be printed out backwards now from right to left.
The new bag will contain " 'l', 'a', 'c', 'i', 'r', 't', 'c', 'e', 'l', 'e' "

lacirtcele

function **addVector**

Here we are going ot be insertring engeineer without reversing it, so the new bag will contain;
" 'l', 'a', 'c', 'i', 'r', 't', 'c', 'e', 'l', 'e', 'e', 'n', 'g', 'i', 'n', 'e', 'e', 'r' ".

lacirtceleengineer

function **removeLastThree**

this time the program iterates through the bag to count all of the elements but it removes the last 3 specific elements that it comes across, so the new bag will look like this since we are removing the passed-in item 'e'.
" 'l', 'a', 'c', 'i', 'r', 't', 'c', 'e', 'l', 'e', 'n', 'g', 'i', 'n', 'r' ".

lacritcelenginr

After cross checking sources I believe my explanation was 100% correct using the data given with the passed-in bag vector and item. 5/5

B.3 – The solution, 25 Points

Please code the **Copy Constructor** (the reverse) and the **removeLastThree** function described in B2 without using the existing functions.

```
Bag(const Bag& b2)
     For(int k = b2.vec.size() – 1; k >= 0; k--){
          vec.push_back(b2.vec[k]);
}}

Void removeLastThree(Char p){
     int a = vec.size();
     int b = 1;
     for (int k = a – 1; k >= 0; k--){
          if (vec[k] == p){
               deleteEl(k); b++;
               if (b == 4)
               break;
}}
```

Not 100% sure if this works as it would need a lot more fine work and tweaking to get running, but as it stands I believe this is the essence of what I would work with to get the copy constructor to work along with the removelastthree function described in B2. 20/25

PART C – 5 Extra Credit Points

Covid-19 and our career in tech.

How are you doing with remote learning? In your opinion, how will what we have been experiencing this semester prepare us for our career in tech? Please provide at least 3 points and briefly discuss them. Thank you.

The more time I spent in remote learning the more I realize how much I don't like it. At first it was nice, not having a commute and being comfortable at home. However everything is much harder to learn. In the long run this might be a good thing, as it pushes me to be independent and to learn things on my own, but overall the difficulty of learning has increased drastically. I think this year is not going to set me up for a good 2021. I will need to review a lot of things in order to get a better understanding if I am ever going to need to use these things we learn on a daily basis in the tech field. I know from a lot of friends who have told me things are much easier in the work force, despite the competition, the learning curve is much lower and you can learn things as you go so it's okay if you don't fully grasp the things you're learning right now, but I'm more concerned about interviews and leaving good impressions. I want to be knowledgeable about my craft. If this remote learning translates into my career field being heavily remotely conducted, then this is all preparing me quite nicely. Though I enjoy a hands on more team oriented physical atmosphere because interaction that way is more impactful to me, this semester is preparing me for a world I was not quite aware would or could exist in my near future.

5/5

END OF EXAM: 5:00 PM

TOTALS:

Part A: 50/50

Part B: 45/50

Part C: 5/0

GRAND TOTAL: 100/100

A

PLEASE DO NOT DETACH THIS PAGE FROM YOUR EXAM.

You may get partial credit for what you write on this page.