CSC 340.03+.04 MIDTERM EXAM 02 SPRING 2022 TA

MIDTERM EXAM INSTRUCTIONS

1. Midterm Exam: **25 points w/ 0 E.C. points**
2. Due Date & Time: **04-25-2021 at 05:00 AM**

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

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	

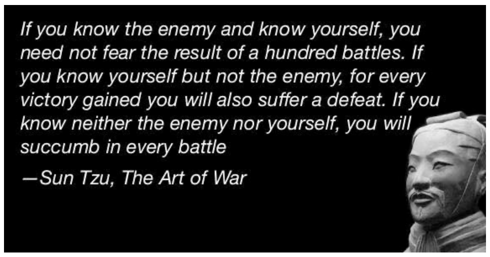
A: 90-100% **B**: 80-89% **C**: 70-79% **D**: 60-69% **F**: 0-60%
The course grader provides feedback to your assignments on iLearn.

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 over the answers to all the exam questions together in class.


If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle
—Sun Tzu, The Art of War

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.

EXAM
Full Name | SFSU ID
- Each question
  - Answer, Step A
  - *Corrected Answer, Step B*
  - *More practice, Step C*
  - *Other notes*

- Next question

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 exam 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 last part of the semester. Thank you.

Kullathon “Mos” Sitthisarnwattanachai
921425216

Step A

- See the following pages for the exam attempts.
- Started: 4/24 07:23 PT
- Finished: 4/24 07:13 PT

Step B

- Correction and evaluations are listed at the end of each questions in a box.
- Total score: 46/100

Question	Scored	Total
A.1	**0**	**10**
A.2	**2**	**10**
A.3	**15**	**16**
A.4	**4**	**4**
A.5	**0**	**10**
B.1	**20**	**20**
B.2	**0**	**5**
B.3	**0**	**15**
C.1	**0**	**5**
C.2	**5**	**5**
Total	**46**	**100**

Step C

- Similar to the last midterm, there are significant amount of code tracing questions in this exam. These questions can take up a lot of time to solve but are also the most valuable ones in the exam. I ended up going for these questions instead, but barely had enough time to attempt other questions.
- This exam was mainly focused on smart pointers. Being a topic that was discussed recently, I thought that it may have been fresh in my memory. Evidently, this exam proved that my understanding of smart pointer have not been nearly as sufficient, and that I have a lot more to revise on the topic.
- Given that code tracing questions are the most rewarding on these exams, I should practice more questions like these in order to speed up my solving skills so that I have time to attempt other questions.

1. Section, Date and Time:
   TIC and TAC, due 04-25-2022 at 05:00 AM

Full Name in **CAPITAL LETTERS** | **SFSU ID**

KULLATHON SITTHISARNWATTANACHAI	921425216

2. Midterm Exam (2 exams, 0 dropped): 100 points
3. To prepare for this exam, please review all the related materials including the 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 papers. No handwriting. No scanned images. No screenshots. 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, closed IDEs, and no communication with anyone except the course instructor. The course instructor will be available on Zoom (zoom.ducta.net) or via email during the exam time. You cannot use any other materials or tools but only the provided exam which will be in Microsoft Word format.
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

---

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.

Step B	
Original score:	*0/10*
Reasoning and corrections:	*Did not attempt. Skipped to attempt higher-valued questions.* *Reference: Package 09* • *Weak pointers are different from other smart pointers in that it does not use reference counting, thus allowing for objects to reference each other while allowing the control block to be destroyed. Other types of smart pointers requires the reference count in their control block to be zero before being destructed.*
Room for improvement:	• *Should have at least written something for the question.*

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?

release() is the most dangerous since it does not delete the object from memory. Failure to capture the address from release() and properly deallocating the memory can lead to memory leak.

Step B	
Original score:	*2/10*

Reasoning and corrections:	*Did not attempt. Skipped to attempt higher-valued questions.* *Only answered the last part of the question.* *Reference: Package 09* • *The last part concerning the use of the release() function was discussed in class.*
Room for improvement:	• *Revise more on the new smart pointer functions.*

A.3 - 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() {
	                                                        // @A = Goofy (creating unique_ptr inline)
	cout << passByValue(make_unique<Name>("Goofy")) << endl;   // "@uPtr_V: @A" // "getName(): Goofy" //  "@A"
	                                                        // (ptr out of scope) "Goofy: destructor called."
	                                                        // @B = Mickey
	unique_ptr<Name> uPtr{ make_unique<Name>("Mickey") };     // (uPtr points to @B, local scope)

	passByRef(uPtr);                                        // "@sPtr_R: @B" // "getName(): Mickey"
	cout << "name_uPtr: " << uPtr << endl;                  //  "name_uPtr: @B"

	passByMove(move(uPtr));                                 // @C = argument is a temp rvalue of uPtr (@B)
	cout << "name_uPtr: " << uPtr << endl;                  // "@uPtr_M: @C" // "getName(): Mickey"
	                                                        // (ptr out of scope) "Mickey: Destructor called."
```

```
                                                                  // "name_uPtr: @C"

    uPtr = make_unique<Name>("Minnie");                           // @D = Mickey
    shared_ptr<Name> sPtr{ uPtr.release() };                      // sPtr now points to @B
    passByShare(sPtr);                                            // "@sPtr_S: @B" // "getName(): Mickey" // "use_count(): 2" (no reset?)


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

How many lines does this program output? **15**

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

See workings above

01	@uPtr_V: @A
02	getName(): Goofy
03	@A
04	Goofy: Destructor called
05	@uPtr_R: @B
06	getName(): Mickey
07	name_uPtr: @B
08	@uPtr_M: @C
09	getName(): Mickey
10	Mickey: Destructor called
11	name_uPtr: @C
12	@sPtr_S: @B
13	getName(): Mickey
14	use_count(): 2
15	END of Program
16	
17	
18	
19	
20	

Step B	
Original score:	*15/16*
Reasoning and corrections:	*Deducted 1 point.* *Reference: Tested the provided code snippet.* • *Line 11 outputs a nullptr value.*
Room for improvement:	• *Review the use of the move() function and what it does. Still not sure on why the pointer has a nullptr address.* • *Side note: it seems that, on some runs, addresses that were thought to be distinct does not appear to be the case. Not quite sure why, though…*

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

To manually destroy an smart pointer block, one could use the release() function and capture its return value, which is the address it is pointing to, assigned to another pointer. For example, if p is a smart pointer pointing to an integer, create a temporary raw pointer tmp as: int *temp{ p.release() }. Because release() does not destroy the object it is pointing to, the address must be captured in order to be set to nullptr manually, i.e. temp = nullptr. Failure to do the latter could lead to a memory leak.

Step B	
Original score:	*4/4*
Reasoning and corrections:	*No deductions.* *Reference: Package 09* • *This was discussed in a recent lecture.*
Room for improvement:	*N/A*

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.**

Step B	
Original score:	*0/10*
Reasoning and corrections:	*Did not attempt. Skipped to attempt higher-valued questions.* *Reference: Tested the provided code snippet.* • *The indirect recursion above will output: 123.*
Room for improvement:	• *Practice more on recursion problems. Tracing recursion functions can be tricky and take a lot of time, so I skipped it to attempt other questions.*

PART B – 40 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
    return shared_ptr<Name>(obj.release());
}

int main() {
    // #2 Insert Code
    auto ptr = func();
    if (ptr) { std::cout << ptr << std::endl; }
    else { std::cout << "Object destroyed." << std::endl; }
}
```

[#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.*

See above.

[#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."

See above.

Step B	
Original score:	*20/20*
Reasoning and corrections:	*No deductions.* *Reference: Tested the provided code snippet.* • *Output works as expected.*
Room for improvement:	• *N/A*

B.2 – The Problem, 5 Points
Similar to what we did in ASMT 4, 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)

function **addVector**

function **removeLastThree**

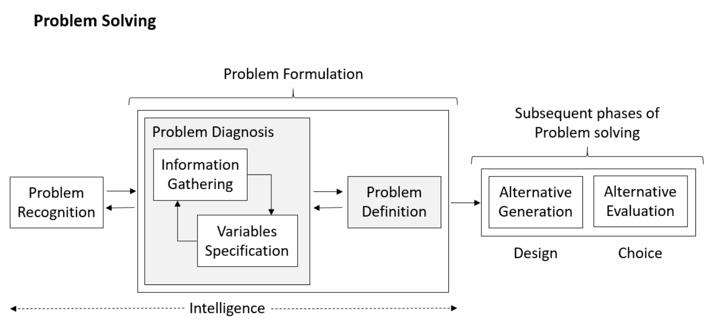
Step B	
Original score:	*0/10*
Reasoning and corrections:	*Did not attempt. Skipped to attempt higher-valued questions.*
Room for improvement:	• *This was not part of the ASMT 04 specification.*

B.3 – The solution, 15 Points
Please code the **Copy Constructor** (the reverse) and the **removeLastThree** function described in B2 without using the existing functions.

Step B	
Original score:	*0/10*
Reasoning and corrections:	*Did not attempt. Skipped to attempt higher-valued questions.*
Room for improvement:	• *This was not part of the ASMT 04 specification.*

PART C – 10 Points

C.1 - 5 pts



What did the IDEO team do during the Problem Formulation phase? What was the most important question? Can we use the same approach in Software Development?

Step B	
Original score:	*0/10*
Reasoning and corrections:	*Did not attempt. Skipped to attempt higher-valued questions.*
Room for improvement:	• *N/A*

C.2 - 5 pts
Your programming team lead/leader asks you to use raw pointer to implement a memory bound function. You know that using Smart Pointers is the way to go. And you heard that the leader does not know Smart Pointers. What would you do and say? Please provide your answer in detail.

Approach the person with respect and suggest the usage of smart pointers. The person may already have knowledge of smart pointers but may not agree with its implementation in a particular use case or may simply did not think to use it. If the previous assumption was correct, explain to them why the use of smart pointers would be advantageous over raw pointers. If they still insist on using raw pointers, it may be of help to continue further discussions with your colleague. Your assumption on the usage of smart pointers may not be accurate. Respectfully acknowledge everyone's point of view and move on.

Step B	
Original score:	*5/5*
Reasoning and corrections:	*Question was largely opinion based.*
Room for improvement:	• *N/A*

PLEASE DO NOT DETACH THIS PAGE FROM YOUR EXAM.
You may get partial credit for what you write on this page.