CS 341 Midterm 1 Feedback

Boyang Chen

TOTAL POINTS

43.5 / 85

QUESTION 1

1 UpdateCity & 0 / 5

✓ **+ 0 pts** **Not assigned this question**

+ 0.5 pts main stack frame

+ 0.5 pts updateCity stack frame

+ 1.5 pts h as household on stack

+ 1 pts h as reference on stack to h from main

+ 0.5 pts string city

+ 1 pts h from main updated with "Springfield"

Deductions

- 0.25 pts Household has wrong form

- 0.5 pts Reference as actual household instead of reference

- 0.5 pts h in heap

- 0.25 pts wrong field in h from main updated with Springfield

- 0.5 pts city in updateCity stack frame not updated with value.

- 0.5 pts city object created in heap

- 0.5 pts city object in updateCity points to heap

- 0.25 pts city object created in main stackframe

- 0.5 pts contents of updatecity in heap

Missing

+ 0 pts no stack frame labels

+ 0 pts no string city in updateCity frame

+ 0 pts h from main not updated to "Springfield"

+ 0 pts missing either main or updateCity stack frame

+ 0 pts h object referencing main object not created in updateCity stack frame

+ 0 pts no answer provided

QUESTION 2

2 UpdateCity 5 / 5

+ 0 pts Not assigned this question

✓ **+ 0.5 pts** **main stack frame**

✓ **+ 0.5 pts** **updateCity stack frame**

✓ **+ 1.5 pts** **h as household on stack**

✓ **+ 1 pts** **h as copy of h from main - this is the important one**

✓ **+ 0.5 pts** **string city**

✓ **+ 1 pts** **h in main not updated with "Springfield"**

Deductions

- 0.5 pts Household has wrong form in main

- 0.5 pts household has wrong form in updatecity

- 0.5 pts h in function as pointer instead of actual household

- 1 pts either h from main or updatecity in heap

- 0.25 pts string city in updatecity is a pointer

- 0.5 pts h in updateCity not updated to Springfield

- 0.5 pts h in updateCity missing values from main

Missing

+ 0 pts no stack frame labels

+ 0 pts missing either main or updateCity stack frame

+ 0 pts h in main updating to springfield

+ 0 pts no string city in updateCity

+ 0 pts no household in updateCity

+ 0 pts no answer given

QUESTION 3

3 LargeImage (addPerson) 0 / 25

+ 1 pts main stack frame

+ 1 pts addPerson stack frame

+ 4 pts Census c

+ 2 pts c.population

+ 2 pts c.places

+ 2 pts Household h

+ 2 pts Person* p

+ **3 pts** &c

+ **2 pts** *h

+ **2 pts** *p

+ **2 pts** updated population

+ **2 pts** updated places

Deductions

- **2 pts** &c as a copy instead of reference

- **1 pts** h exists not pointing to anything/pointing to wrong item

- **1 pts** p exists not pointing to anything/pointing to wrong item

- **1 pts** c incorrect form (-2?)

- **1 pts** vector has the wrong form

- **4 pts** All objects in the heap instead of on the stack

- **2 pts** vector of pointers instead of objects

Missing

+ **0 pts** no stack frame labels

+ **0 pts** missing either main or addPerson stack frame label

+ **0 pts** missing c

+ **0 pts** missing c.population

+ **0 pts** missing c.places

+ **0 pts** didn't update c.population

+ **0 pts** didn't update c.places.

+ **0 pts** missing *p

+ **0 pts** missing *h

+ **0 pts** missing &c

+ **0 pts** missing Person *p

+ **0 pts** missing h

✓ **+ 0 pts no answer provided**

QUESTION 4

4 Construct a test case 2.5 / 3

+ **0.5 pts** Create a census object

✓ **+ 0.5 pts Adds at least one Person to the population vector**

✓ **+ 1 pts Calls meanAge**

✓ **+ 1 pts Compares against the mean age of the inputs provided**

Missing

✓ **+ 0 pts Missing census object**

+ **0 pts** Missing population vector

+ **0 pts** Missing meanAge call

+ **0 pts** Missing mean to test against

Incorrect

+ **0 pts** No answer

+ **0 pts** different test case types too vague

+ **0.5 pts** REQUIRE(abs(S.Exam() - 80.5) < 0.001) - mimiced samples/concepts used in class but doesn't apply them to the current question.

+ **0 pts** Negative values should fail the test

+ **0.5 pts** Manually recalculates average by adding elements of "vector", then dividing by size. - Independent verification of testing does something similar, but we want to test against constants, not reimplement the code.

+ **0 pts** Checks if the vector is not null using assertArrayEquals to check if expected values would match actual vector

+ **0 pts** Use input numbers and check if the outcome is different. - Student takes the average of 7, 3, 2, 1

+ **0 pts** Populates people vector and check if the average returned is correct - Doesn't specify meanAge call or the mean to test against

+ **0 pts** Check if average meets a certain threshold (< 0.0001)

+ **0 pts** Student says the result should be 30, but the average of the inputs given (99, -1, 22, 0) is 60.5. Unsure of where 30 is coming from.

+ **0 pts** Empty test case, but creates census object.

+ **0 pts** Pushes ages onto a stack and stores the total age and divides it by total population of the stack. Mentions if the average age was above at least 18 would return true, false is below.

+ **0 pts** Used array instead of vector

QUESTION 5

5 Call accumulate 4 / 5

+ **0 pts** Didn't have question

+ **0 pts** Didn't call function

✓ + **0 pts** **accumulate(**

first parameter (begin)

+ **0.5 pts** C.begin()
+ **0 pts** C.start()
+ **0 pts** V.begin()
+ **0 pts** begin
+ **0 pts** o.begin() - initialized vector o;
+ **0 pts** first
+ **0.25 pts** container.begin()

second parameter (end)

+ **0.5 pts** C.end()
+ **0 pts** V.end()
+ **0 pts** end
+ **0 pts** o.end() - initialized vector o;
+ **0 pts** last
+ **0.25 pts** container.end()

third parameter (init)

+ **1 pts** 0
+ **1 pts** T{}
+ **0 pts** C
+ **0 pts** init - undefined variable
+ **0 pts** not included
+ **0 pts** 1

fourth parameter header

+ **1 pts** \[] (const auto &result, const auto &b)
+ **1 pts** (T x, T y)
+ **0 pts** (T elem) - single parameter
+ **0.5 pts** (int sum, int val)
+ **0.25 pts** []
+ **0.25 pts** auto
+ **0.25 pts** const
+ **0.25 pts** &
+ **0 pts** not const
+ **0 pts** no &
- **0.25 pts** [=]
- **0.25 pts** [&]
+ **0 pts** None

fourth parameter body (sumIfPos)

+ **1 pts** if (y>0) return x + y;
+ **1 pts** else return x;
+ **0 pts** return x + y; - added all elements regardless if positive
+ **0 pts** return val > 0; - param to filter, not accumulate
+ **0 pts** if (elem > 0) return true; - returns incorrect value inside if case (just elem, boolean, etc.)
+ **0 pts** return false; - returns incorrect value outside if case (0, boolean, etc.)
+ **0 pts** a+=b - doesn't return anything
+ **0 pts** missing return outside of if condition
+ **0 pts** None
+ **0 pts** if((va1 >0) && (val2 > 0)) {return val1 + val2;}
if((val > 0) && (val2 <=0)) {return val1;}
if((val <= 0) && (val2 >0)) {return val2;}
else {return 0;} })
+ **0 pts** {if(val>0) return val;} -- no return value in the else case, only one parameter provided in solution

Incorrect/alternative calling parameters

+ **0 pts** Uses container name other than C.
+ **0 pts** missing third parameter init
+ **0 pts** [](const T& elem) - incorrect single parameter
+ **0 pts** return elem>0; - parameter to filter, not accumulate
+ **1 pts** T{} (instead of 0) - default may not be "0" for types being summed
+ **1 pts** \[](const auto &result, const auto &b)
+ **0 pts** Used C.start() instead C.begin()
+ **0 pts** Tested for <0, different behavior when =0

Incorrect

+ **0 pts** Implemented accumulate instead of calling it.
+ **0 pts** Tried to compute accumulate without calling or defining function
+ **0 pts** Added all the elements, regardless if positive or not
+ **0 pts** No answer provided

Implementation of accumulate first param (begin)

+ **0 pts** Max of 2, split points as you deem appropriate
+ **0 pts** loop condition .5, if condition .5, sum operation .5, no init+return .5
+ **0 pts** T first - template variable
+ **0 pts** InputIt first - undefined
+ **0 pts** C first - C is a specific object, not a type
+ **0 pts** C:: iterator.begin()
+ **0 pts** T.begin()
+ **0 pts** None

Implementation of accumulate second param (end)

+ **0 pts** T last - Template variable
+ **0 pts** InputIt last - Undefined variable
+ **0 pts** C last - C is specific container, not a type
+ **0 pts** C::iterator.end()
+ **0 pts** None
+ **0 pts** T.end()

Implementation of accumulate third param (init)

+ **0 pts** T init
+ **0 pts** T sum
+ **0 pts** None

Implementation of accumulate fourth param header

+ **0 pts** BinaryOperation op
+ **0 pts** None

Implementation of accumulate fourth param body (sumIfPos)

+ **0 pts** while (first != last) {
if ((*first) > 0) {
init = op(init, (*first));
}
first++;
}
return init;
+ **0 pts** for (first : last) {
if (*first > 0) {
sum += *first;
first++;
}
}
+ **0 pts** while (first != last) {
if ((*first) > 0)
init += (*first);
++first;
}
return init;
+ **0 pts** add = 0;
while (begin != end) {
if (F(*begin > 0) {
add += (*begin);
}
++begin;
}
return add;
+ **0 pts** while(C.begin() != C.end()) {
init=op(init,*first);
++first;
}
return init;
+ **0 pts** for (; first != last; first++){
init = op(init, *first);
}
return init``
+ **0 pts** while(first != last) {
if(first > 0 && last > 0) {
init = op(init, *first);
}
++first;
}
return init;
+ **0 pts** auto E1 = *(C.begin());
auto E2 = *(C.end());
int sum = init;
for(auto i : C){
if ((E1 <= i) && (i < E2)){ sum += i;
}cout << sum << endl;``
+ **0 pts** T initial_value = init;
while (rst ! = last) {
initial_value = op (initial_value, *rst);
++rst;
}

return initial_value;

+ **0 pts** for

(Customer& c : T) {

cout <<

c.sum() << endl

Calling accumulate (begins with "0")

✓ + **0.5 pts** **C.begin()**

✓ + **0.5 pts** **C.end()**

✓ + **1 pts** **0**

+ **1 pts** \[](T x, U y)

✓ + **1 pts** **{ if(y>0) return x+y;**

✓ + **1 pts** **else return x; });**

QUESTION 6

6 range-for sum pos 0 / 5

✓ + **0 pts** **Problem wasn't given**

initialize sum

+ **1 pts** int sum = 0

+ **1 pts** T sum

+ **1 pts** auto sum = T{};

+ **0 pts** Didn't initialize sum variable

Range-based for loop

+ **2 pts** for(T &val : C)

+ **0 pts** T

+ **0 pts** auto

+ **0 pts** &

+ **0 pts** no &

+ **0 pts** const

+ **0 pts** no const

+ **1 pts** for(C : element)

+ **0.5 pts** for(auto begin : end)

- **1 pts** for(auto i = C.begin(); i != C.end(); ++i)

- **1 pts** for(int i = 0; i<C.size(); ++i)

- **1 pts** while(b!=e)

Check for pos/neg

+ **1 pts** if(val>0)

+ **0 pts** doesn't check

Compute sum

+ **1 pts** sum+=val;

+ **0 pts** sum = val

QUESTION 7

7 Update to equivalent lambda 5 / 5

+ **0 pts** Wasn't asked

Capture

+ **1.5 pts** [&]

+ **1.5 pts** [=]

✓ + **1.5 pts** **[city]**

+ **1.5 pts** [&city]

+ **0 pts** []

+ **0 pts** no capture semantics included

Parameters

✓ + **1 pts** **Did not include city as a parameter (because the question asks you not to).**

✓ + **1.5 pts** **Household& h**

+ **0.5 pts** auto& h

+ **0.5 pts** Household h

+ **0 pts** auto h

+ **0 pts** type not specified

+ **0 pts** Household not given

- **0.5 pts** const on Household

+ **0 pts** string city

+ **0 pts** Household h.city

body

✓ + **1 pts** **h.city = city**

+ **0 pts** h = city

+ **0 pts** h.city.push_back(city)

+ **0 pts** city

+ **0 pts** city = newCity

+ **0 pts** No body

+ **0 pts** Wasn't answered

Incorrect parts

+ **0 pts** Gave a name to the lambda

+ **0 pts** bad syntax

+ **0 pts** -> city

+ **0 pts** Incomplete

- **0.5 pts** Extra city = "Springfield" in function

- **0.5 pts** returns true for a void function

- **1 pts** void updateCity at the beginning, same

notation as function with extra [] stuck between name and parameters

- 0.5 pts return h - function should return void

QUESTION 8

8 Implement destructor 2 / 5

✓ **+ 2.5 pts loop through population, delete on each element**

✓ **+ 2.5 pts loop through places, delete on each element**

+ 0 pts No answer given

Partial Points

+ 1 pts deleted on variable p and h, didn't loop through vectors

+ 2 pts Iterated through an unknown declared vector, but called delete on each element

Deductions

- 1 pts used incorrect type in for each loop. Ex: int, float,

+ 0 pts auto *ptr instead of auto ptr

- 0.5 pts (auto vector: ptr) instead of (auto ptr: vector)

- 1 pts used "delete *ptr" instead of "delete ptr"

- 1 pts used delete[] instead of delete

- 2 pts Set vector[index] to nullptr then deleted

- 1 pts Used vector[pointer] instead of vector[index]

- 1 pts Used vector::at(pointer) instead of vector::at(position)

- 3 pts calls vector::back (which just deletes the last element)

✓ **- 3 pts called delete on vector(s)/class**

Feedback

+ 0 pts No loops, no delete

+ 0 pts Only declared destructor, no implementation given

+ 1 pts Created a pointer then deleted

QUESTION 9

9 applyBinPos/Neg 8 / 10

✓ **+ 1 pts ValType total;**

+ 3 pts Set total to first pos/neg element

+ 1 pts while(*begin>/<0)

✓ **+ 1 pts while(begin!=end)**

✓ **+ 1 pts if(*begin>/<0)**

+ 2 pts total = op(total, *begin)

✓ **+ 1 pts ++begin**

✓ **+ 1 pts return total**

+ 0 pts Just called function/No partial points possible

+ 0 pts No answer given

Deduction

+ 0 pts Didn't use given parameters

+ 0 pts total is inside for loop

+ 0 pts returned a vector of the filtered list

+ 0 pts && begin!=end

+ 0 pts op(*begin, *end)

+ 0 pts Used begin to get value instead of *begin

Partial Points

✓ **+ 1 pts begin total equal to 0 instead of first pos or neg element**

+ 1 pts Set total to default constructor when it should be the first pos/neg element

+ 2 pts initialize total to first element without checking if pos/neg

✓ **+ 1 pts total += *begin**

+ 1 pts total += op(total, *begin)

+ 0.5 pts total = op(total, *(++begin))

+ 2 pts op(total, *begin)

✓ **+ 1 pts total = f(a,b) but a and b were the wrong values passed in.**

+ 0 pts called op(total, *begin) but did set it equal to total

Feedback

+ 0 pts Type of total to something other than ValType

+ 0 pts Does pairwise comparison on current and next element instead of current element

+ 0 pts Just calls accumulate

QUESTION 10

10 Learning more than one language 6 / 6

General programming language benefits

✓ **+ 3 pts** **Flexibility of thought**

✓ **+ 3 pts** **Have the right tool for the right occasion**

+ 3 pts Understand different methods of problem solving

+ 3 pts Open the door to more career options

✓ **+ 3 pts** **Contrast grants insight into programming language details**

+ 3 pts C++ has common features with other programming languages (F# static, Deterministic Resource C, OO Java)

+ 3 pts Some libraries/clients are language exclusive

+ 3 pts Language features from other languages can be used to evolve the language itself

+ 3 pts Reasoning for correctness, parallel programming may be easier in C++

Challenging to justify

+ 2 pts Easier to understand terminology

+ 1 pts learning more languages makes you more efficient - how? can do efficient test and enforce the performance of code - doesn't explain

+ 1 pts Python as a dynamic language offers types at runtime and types of errors on execution - these are worse than what C++ offers, types at compile time and errors at compile time.

- 1 pts C++ uses more memory

+ 0 pts Don't need to create a new file for a class like you do in java

+ 1 pts C++ is more productive in terms of GUI and game programming - depends on the type of game and which libraries/engines being used. Also by what you mean by productive, efficient is why C++ is used, but getting a game running will generally be faster in Python

+ 0 pts C++ is a dynamic programming language - incorrect

+ 1 pts C++ has OOP

+ 1 pts Only one benefit

+ 0 pts No answer

QUESTION 11

11 Differences between C and C++ 3 / 3

+ 1 pts Object oriented

+ 1 pts classes

+ 1 pts templates

+ 1 pts Iterators

+ 1 pts lambdas

+ 1 pts range-based for loops

+ 1 pts parametric polymorphism

+ 0 pts Wrong format

+ 1 pts References/pass by reference

+ 1 pts STL

✓ **+ 1 pts** **Higher level abstractions**

+ 1 pts Overloading

+ 1 pts Inheritance and encapsulation

Memory management

+ 1 pts new() and delete()

✓ **+ 1 pts** **RAII**

✓ **+ 1 pts** **No explicit memory management. - there is, but we try to bury this behavior in class implementation**

+ 0 pts Deterministic resource management - this is a feature the languages share

+ 1 pts no artificial limits - this isn't totally true, but presumably you were referring to how arrays need a static upper bound on size, while vectors do not.

+ 1 pts Exception handling

+ 1 pts Vectors

+ 1 pts Namespace

+ 1 pts In line functions

+ 1 pts Scope resolution operator

+ 1 pts Thread

+ 1 pts fstream

+ 1 pts Constructor and destructors

+ 1 pts Generic Types

+ 1 pts string.h

+ 1 pts smart pointers

+ 1 pts direct initialization

+ 1 pts Use of namespaces

+ **1 pts** cin/cout instead of scanf printf - C++ offers access to both

+ **1 pts** No pointers in C++, but there are pointers in C - we try to avoid using them externally to classes, they are still used internally

+ **0 pts** C++ can create objects on the stack, and pointers to objects on the stack - C can do this as well

+ **0 pts** User defined data types - structs exist in both C and C++

+ **0 pts** Wrong/No Answer

QUESTION 12

12 iterator vs reference 4 / 4

✓ **+ 2 pts iterators allowed to be null**

+ **2 pts** iterators can traverse data structures

+ **2 pts** end() represents no value returned

+ **2 pts** Arithmetics with iterator

+ **2 pts** Change/access value of item in data structure

+ **2 pts** Use on templates

+ **2 pts** Re-referencing allowed

✓ **+ 2 pts Generic usage - usable with STL algorithms**

+ **2 pts** Can be used in forward and reverse traversals

+ **2 pts** Const iterator allows you to access a value while preventing changes

+ **1 pts** Access first element with begin() - property of container, not iterator

+ **0 pts** Wrong/No Answer

Incorrect

+ **0 pts** Iterators are unable to change the data they refer to while references can - false

+ **1 pts** Iterators can find beginning and end of even unsorted data

+ **1 pts** To use in a range-based for each loop - the iterators are invisible in this data structure, you don't have to choose between the two

+ **0 pts** Iterators will never point to null - false

+ **0.5 pts** Iterator is safer than reference - in what way?

+ **0.5 pts** iterator is easier to use than reference because it has functions inside - what functions would you want to be internal to a reference, it offers transparent access to the underlying variable

+ **0.5 pts** Will not change data when it accesses memory - neither does a reference

+ **0.5 pts** To return an object - this is a feature that iterators and pointers have in common

+ **0 pts** iterators return after each iteration - what?

+ **0.5 pts** End points directly to an element - end points outside the bounds of the container

+ **0 pts** Less resource heavy. - iterators potentially use more resources, at least in memory if not via access, to keep track of the current location in the container.

+ **0.5 pts** Removes the need for you to keep track for a memory leak - not needed with references either

+ **0.5 pts** Iterators can access objects on the heap - so can references

QUESTION 13

13 reference vs pointer 4 / 4

+ **0 pts** Question was not assigned

+ **2 pts** references always valid, pointer arithmetic can generate invalid locations

✓ **+ 2 pts references guaranteed to refer to something, pointers can be null, do not need to be initialized when declared**

✓ **+ 2 pts references always refer to the same thing, pointers can be reassigned**

✓ **+ 1 pts At least 2 reasonable reasons**

+ **2 pts** Avoids memory leaks/invalid accesses not knowing when delete is being called on a pointer

+ **2 pts** Pointer has to be dereferenced every time when used

+ **2 pts** Use like a variable

✓ **+ 1 pts Pass by reference**

+ **1 pts** Reference easier to manage/use - why

+ **1 pts** Safer - why?

+ **0 pts** Wrong/No Answer

Incorrect

+ **0 pts** References allow us to overload operators

- 1 pts Pointers use more memory compared to references

- 1 pts References let you change a value from another scope while pointers do not

+ 0 pts Answered iterators vs pointers instead of references vs pointers

Grade Test: Midterm 1 2 pm

*Assign a grade and feedback for the current test attempt. Expand the **Test Information** section to clear the student's attempt or edit the test.*
More Help

Jump to... Hide User Names

Viewing 15 of 63 gradable items

Boyang Chen (Attempt 1 of 1)

Test Information

QUESTION 1: TRUE/FALSE **0 out of 0 points**

These class definitions for a Census application will be used during the test

```
class Household
{
public:
    int numPeople;
    string city;
    string state;
public:
    Household()
        : numPeople(0), city("N/A"), state("N/A")
    {}

    Household(int n, string c, string s)
        : numPeople(n), city(c), state(s)
    {}


};
```

```
class Person
{
private:
    int age;
    bool employed;
public:
    Person(int a)
        : age(a), employed(false) {}
    Person(int a, bool emp)
        : age(a), employed(emp) {}

    int getAge()
    {   return age; }
    int getEmp()
    {   return employed;    }

};
```

```
class Census
{
public:
    vector<Person> population;
    vector<Household> places;
public:
    Census() {}
    void pollHousehold(string city, string state);
    int numPeople();
    int numPlaces();
    double meanAge();
    double meanResidency();

};
```

There is no answer to this question, I chose this question type because it took up less screen real estate than the others I tried out.

Given Answer: True
Correct Answer: True

QUESTION 2: MATCHING **9 out of 10 points**

Determine if the following statements describe C, C++, or both:

Question	Correct Match	Given Match
OOP is built in; size and memory layout of objects is deterministic.	B. C++	B. C++

Uses structures	C. Both	C. Both
Manages resources for you	B. C++	B. C++
No automatic garbage collection	C. Both	C. Both
Higher-level abstractions	B. C++	B. C++
OOP is not built in; Lacks the ability to declare encapsulation.	A. C	A. C
Static	C. Both	C. Both
Uses classes	B. C++	B. C++
Procedural	C. Both	A. C
Uses the STL	B. C++	B. C++

QUESTION 3: MATCHING **7.99999 out of 9 points**

Determine if the following statements describe references or pointers:

Question	Correct Match	Given Match
Must be initialized when declared	A. References	A. References
Needs dereferencing operator to access value	B. Pointers	B. Pointers
Can be used like normal variables	A. References	A. References
“&” must be used at time of declaration	A. References	A. References
has its own memory address and size on the stack	B. Pointers	B. Pointers
often made NULL to indicate that they are not pointing to any valid thing	B. Pointers	B. Pointers
once created, cannot be later made to reference another object	A. References	A. References
Can be declared without initialization	B. Pointers	B. Pointers

shares the same memory address with the original variable but also takes up some space on the stack	A. References	B. Pointers

QUESTION 4: SHORT ANSWER - out of 3 points

Suppose I wanted to test the that function meanAge from the Census class is correctly computing the average of the age of the Person objects in the Population vector.

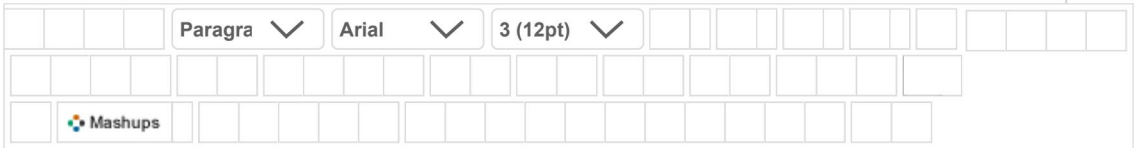
How would you test this function? Describe a single test case by the inputs you would provide, and the expression which would return true if the function passes the test, and false if it does not.

Note: This function has undefined behavior if there are no people in the Population vector.

Given Answer:
// Inputs provided: 3 Persons with ages: 10, 10, 25, 8
# inlcude<cmath>
REQUIRE(abs(meanAge() - 13.25) < 0.0001)

Correct Answer: [None]

Response Feedback:



QUESTION 5: FILE RESPONSE - out of 5 points

Using the previously defined Household class, evaluate the following code.

```
void updateCity(Household h, string city)
{
    h.city = city;
    // POINT A
}

int main()
{
    Household h(1, "Detroit", "Illinois");
    updateCity(h, "Springfield");
    return 0;
}
```

Draw the state of memory (stack and heap) when execution reaches **point A** --- draw your answer on paper and take a picture or digitally create an image. Label each stack frame (with name of function), label variables and parameters, and include the contents of each memory cell if known; draw "?" if the contents are unknown.

Given Answer: updatecity.jpg

Response Feedback:

Paragra Arial 3 (12pt)

Mashups

QUESTION 6: SHORT ANSWER - **out of 5 points**

Suppose I have some STL container C, which has begin and end implemented with a Forward Iterator.

Show by writing the code how you would use std::accumulate to get the sum of only the positive elements (>0) in the container.

accumulate has the following declaration
template< class InputIt, class T, class BinaryOperation >
T accumulate(InputIt first, InputIt last, T init, BinaryOperation op);

And is defined to do the following
Computes the sum of the given value init and the elements in the range [first, last) using the given binary function op.

Given Answer: std::accumulate(C.begin(), C.end(), 0, [ ](int a, int b){return (b > 0) ? a+b: a;});

Correct Answer: [None]

Response Feedback:

Paragra Arial 3 (12pt)

Mashups

QUESTION 7: SHORT ANSWER

- out of 5 points

Suppose I have the following function.

```
void updateCity(Household& h, string city)
{
    h.city = city;
}
```

Convert it to an equivalent lambda expression, which only takes a single parameter of the household to be updated, and instead captures the city to update via a closure.

Given Answer:
[city] (Household& h) {
h.city = city
}

Correct Answer: [None]

Response Feedback:

Paragra Arial 3 (12pt)

Mashups

QUESTION 8: ESSAY

- out of 10 points

Suppose I wanted to add a function to the algorithms library, which given the beginning and ending iterators and a binary function to apply, applies the function pairwise to elements, but only for elements which evaluate as positive (>0).

I've included the declaration of the function below, you must provide the implementation matching the behavior described above.

template <typename Iter, typename ValType, typename Func>
ValType applyBinPos(Iter begin, Iter end, Func f)

For example, the following code calling the function

```
vector<double> V;
V.push_back(3.14);
V.push_back(-13.3);
V.push_back(-12.45);
V.push_back(100.0);
V.push_back(20.31);

double total = applyBinPos
    <vector<double>::iterator, double, std::function<double(double, double)>>
    (V.begin(), V.end(),
    [](double a, double b) {return a + b; });

cout << total << endl;
```

Produces the output 123.45. The binary function is not guaranteed to be +.

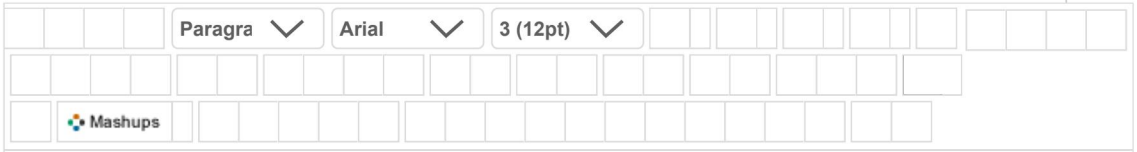
You can assume that the container has at least one positive value in it.

Given Answer:

```
template <typename Iter, typename ValType, typename Func>
ValType applyBinPos(Iter begin, Iter end, Func f) {
  vector<ValType> pos_pairs;
  for (Iter it = begin();  it != end(); ++it ) {
    if (i > 0)
      pos_pairs .push_back(i);
  }
  ValType result = 0;
  for (Iter it = pos_pairs.begin(); it != pos_pairs.end(); ++it) {
    ValType curr = *(it+1);
    ValType prev = *it;
    result += f(prev, curr);
  }
  return result;
}
```

Correct Answer: [None]

Response Feedback:



QUESTION 9: MULTIPLE CHOICE **2 out of 2 points**

A _____ takes types and sometimes values as input, and produces classes and functions as output.

Given Answer: template

Correct Answer: template

QUESTION 10: MULTIPLE CHOICE **2 out of 2 points**

In C++ when you dynamically allocate an object with new, that memory allocation exists until the that object is deallocated by calling delete on a pointer to that object, at which point the memory is immediately freed.

This is a result of C++ having which property?

Given Answer: C++ has deterministic memory management

Correct Answer: C++ has deterministic memory management

QUESTION 11: CALCULATED NUMERIC **0 out of 2 points**

Given the following definition of a template

```
template <typename T, typename U>
bool DoStuff(T a, U b, std::function<T(U)> f)
{
    return a == f(b);
}
```

With the following calls (explicit typing of templates omitted), how many instances of the function DoStuff are created?

```
template <typename T>
int floor_It(T val)
{
        return (int)floor(val);
}

auto a = DoStuff(1, 1, [](int x) {return x; });
auto b = DoStuff(3, 3.14, floor_It<double>);
auto c = DoStuff(1, "1b", [](string s) {return s[0] - '0'; });
auto d = DoStuff(1, -1, [](int x) {return -x; });
auto e = DoStuff(1, 'b', [](char c) {return c - 'a'; });
auto f = DoStuff(0, 92.5, [](double x){ if(x<0) return -1;
                                        if(x>100) return 1;
                                        return 0; });
```

Given Answer: 5

Correct Answer: 4

Answer range +/- 0 (4 - 4)

QUESTION 12: SHORT ANSWER **- out of 5 points**

Suppose that instead of Census, I used the following definition for the class CensusPointers

```
class CensusPointers
{
public:
    vector<Person*> population;
    vector<Household*> places;
public:
    CensusPointers() {}
    void pollHousehold(string city, string state);
    int numPeople();
    int numPlaces();
    double meanAge();
    double meanResidency();
    void addPerson(int age, bool emp)
    {
        Person* p = new Person(age, emp);
        population.push_back(p);
    }
    void addHousehold(int num, string city, string state)
    {
        Household* h = new Household(num, city, state);
        places.push_back(h);
    }
    ~CensusPointers();

};
```

Provide an implementation of the destructor for CensusPointers which would prevent this class from producing memory leaks.

Given Answer:

```
~CensusPointers() {
  for (auto p: population) {
    delete p;
  }

  for (auto p: places) {
    delete p;
  }
  delete population;
  delete places;
}
```

Correct Answer: [None]

Response Feedback:

QUESTION 13: FILE RESPONSE - out of 25 points

Using the provided definitions for Person, Household, and Person:

```
void addPerson(Census& c, Household* h, Person* p)
{
    c.population.push_back(*p);
    c.places.push_back(*h);
    // POINT A
}

int main()
{
    Census c;
    Household h(0, "Chicago", "Illinois");
    Person* p = new Person(25,true);

    addPerson(c, &h, p);
    return 0;
}
```

Draw the state of memory (stack and heap) when execution reaches **point A** --- draw your answer on paper and take a picture or digitally create an image. Label each stack frame (with name of function), label variables and parameters, and include the contents of each memory cell if known; draw "?" if the contents are unknown.

[For **std::vector**, assume an implementation similar to that discussed in class, with 3 values: **A**rray, **S**ize, and **C**apacity. Also assume that when a std::vector is first constructed, an underlying array of capacity 4 is created. Finally, assume that std::vector provides a copy constructor that performs a deep copy if necessary.]

Given Answer: [None Given]

Response Feedback: Paragra Arial 3 (12pt) Mashups

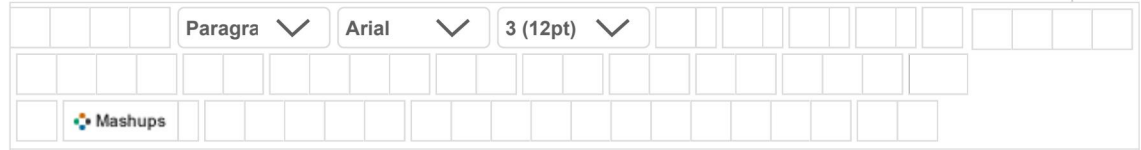
QUESTION 14: SHORT ANSWER - out of 6 points

What benefits are there to learning more than one programming language?

Given Answer: Programming languages influence how you think. Each language has different features like C++ has references but C does not.
Some languages provide high level abstraction like Python which might be useful when you don't want to deal with low-level stuff like memory management, pointer arithematic and stuff and hence write easy to understand code. While in other cases, when writing for high performance, you may need to use the low-level stuff like in C++. Also each language has different use cases and features better suited for a particular task. Like F# for functional progamming, and C++ for OOP, Pyhton for data science and so on.

Correct Answer: [None]

Response Feedback:



QUESTION 15: SHORT ANSWER - out of 4 points

Why might you want to use an iterator instead of a reference?

List two reasons.

Given Answer: 1. Iterator can point to nullptr which means we can use it to point to the end (one element past the last) of a container useful in range based loops.

2. it is generic which means it allows code reuse. An STL just have to implement an interator and the auto and range based loop will work.

Correct Answer: [None]

Response Feedback:

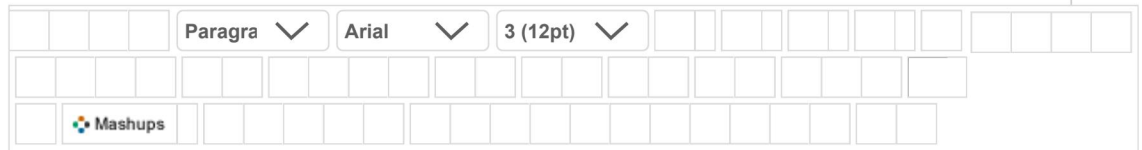
QUESTION 16: SHORT ANSWER - out of 4 points

What advantages do references offer over pointers?

Given Answer:
References can not be set to nullptr so that avoids nullptr access errors.
References are also easier to use we dont need to explicitly dereference before using them as compared to pointers.
Can be used in pass-by-reference for passing values to functions in C++.
References are always initialized and it cannot be changed to refer to another variable therefore avoiding any logical errors that we see with pointers being mistakenly reassigned.

Correct Answer: [None]

Response Feedback:



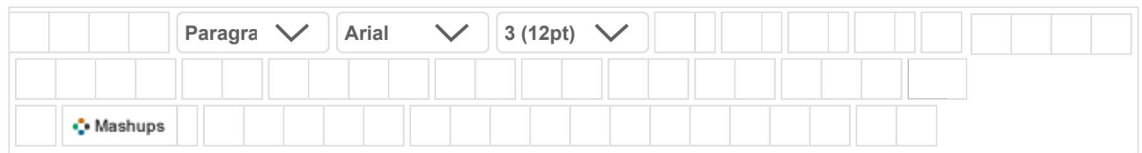
QUESTION 17: SHORT ANSWER - out of 3 points

What are some of the differences between modern C++ and C? List at least 3 things that you will find in C++ code but not C code. Label your list with numbers (1. item 1, 2. item 2, 3. item 3, …)

Given Answer:
1. No pointers.
2. No explicit memory management
3. Higher level abstractions
4. Resources automatically disposed

Correct Answer: [None]

Response Feedback:



FEEDBACK AND NOTES FOR ATTEMPT

Feedback to Learner

Grading Notes

Exit | Save and Exit | Save and Next