CSE 332S: Fall 2019

Name:	
Student ID:	
Date:	

You will have 120 minutes for this exam. **Do not open the exam or start until the designated start time (6:30 PM).** You may begin to look through the provided code if you would like.

As a reminder:

- The exam is closed book and no electronics are allowed
- You are not allowed to collaborate with your neighbors

For this exam, you will use the supplied code to answer questions. The files that should be supplied to you are below. Beside each file name you will find the problems that require it:

File Name	**When to use?**
myvect.h, myvect.cpp	Problem 3
containerstats.h, containerstats.cpp, myvectwithstats.h, myvectwithstats.cpp	Problem 4, 5

Problem #	Points possible	Points earned
1	6	
2	12	
3	18	
4	10	
5	10	
Extra credit	(2)	
Total	56(58)	

Problem 1. (6 points) Given the following functions, answer the questions by giving the output produced by the supplied main function at the specified line. If the program produces an undefined output or error, say so and tell why. Assume all necessary headers are included and the std namespace is being used. If a line produces undefined output or an error, continue on to the next question as if the line causing the error was commented out. For each question, give only the output produced by the lines of code in question.

```
void foo1 ( int i, int * const j, const int &k ) {
      *j = i;
      i += k;
      i = *j + k;
}

int & foo2 (int i) {
      int x = 10 * i;
      return x;
}

void foo3 ( char *& s ) {
      char first = *s;
      cout << first << endl;
      ++s;
}

int main ( int argc, char * argv[] ) {
      int a = 2;
      int b = 5;
      int c = 7;
      foo1 (a, &b, c);
      cout << a << " " << b <<  " " << c << endl;                    // 1

      int &p = foo2(5);
      cout << p << endl;                                             // 2

      char * d = "hello";
      foo3(d);                                                       // 3
      cout << *d << endl;                                            // 4
      return 0;
}
```

a) (3 points) Give the output (or describe the error or errors if there are multiple) produced by the line marked 1 in a comment.

b) (1 point) Give the output (or describe the error) produced by the line marked 2.

c) (2 points) Give the output (or describe the error) produced by the lines marked 3 and 4.

Problem 2. (12 points) For this problem, you will be given snippets of code and asked questions about those snippets. You should assume we are using namespace std and all necessary header files are included.

Each of the code snippets below contains either a compile time error, a potential runtime error, an unsafe operation, or contains no error and produces output**.** For each code snippet:

- If the code contains an error or unsafe operation, place the letter associated with the error in the table below into the output column for that snippet. Errors may be used more than once. An error does not have to be used. Any output produced before the error occurred can be ignored.

- If the code does not contain an error, give its output.

Possible errors:

A. Can't alias a const variable via a reference or pointer to a non-const type	B. illegal/unsafe pointer dereference
C. Illegal pointer arithmetic	D. Uninitialized reference
E. Can't modify a const variable	F. Use of a reference or pointer to an object that has been destroyed
G. Memory leak	H. Double deletion
I. Operator or function is not defined for the given type	J. Indexing out of the bounds of a container, behavior is undefined
K. Exception propagated out of main() uncaught	L. Ambiguous function call

Code:	Output:
`forward_list<int> li;` `li.push_front(2);` `li.push_front(3);` `li.push_front(4);` `cout << li[2] << endl;`	
`void func(vector<int>::iterator it) {` `  it += 2;` `  return;` `}` `int main(int argc, char * argv[]) {` `  vector<int> v = { 0, 1, 2, 3, 4 };` `  auto it = v.begin();` `  func(it);` `  cout << *it << endl;` `  return 0;` `}`	
`int * func() {` `  int j = 10;` `  int * p = new int(j);` `  return p;` `}` `int main(int argc, char *argv[]) {` `  int * p = func();` `  cout << *p << endl;` `  delete p;` `  return 0;` `}`	
`int arr[5] = { 1,2,3,4,5 };` `int i;` `int *p = arr + i;` `cout << *p << endl;`	

`int i = 1;` `const int j = 50;` `int * p = &i;` `*p = 50;` `p = &j;` `cout << *p << endl;`	
`int i = 5;` `int j = 20;` `int &k = j;` `int *m = &i;` `k = *m;` `cout << “i:” << i << “ j:” << j <<` `endl;`	
`int main(int argc, char *argv[]) {` `  int * p = new int(10);` `  int & r = *p;` `  delete p;` `  p = nullptr;` `  cout << r << endl;` `}`	
`char *c = “cse332”;` `char *e = “ exam”;` `char *concat = c + e;` `cout << concat << endl;`	
`int arr[] = {1,2,3,4,5};` `int *p = &arr[3];` `int *q = arr + 1;` `int distance = p - q;` `int difference = *p - *q;` `cout << "dist: " << distance << "` `diff: " << difference << endl;`	

`` int i = 2; int j = 3; int k = 4; vector<int> v; v.push_back(i); v.push_back(j); v.push_back(k); v[0] = 15; v[1] = 16; v[2] = 17; cout << "i:" << i << " j:" << j << " k:" << k; ```	
`` short bar(short i) { return i; } long bar(long i) { return i; } int main(int argc, char * argv[]) { char c = 'c'; cout << bar(c) << endl; } ```	
`` int arr[5] = { 1,2,3,4,5 }; int * const p = arr + 2; cout << *p << endl; ++p; cout << *p << endl; ```	

Problem 3. (18 points) Using the code provided(myvect.h, myvect.cpp) and the main function below, answer the following questions.

- Assume we are **using the std namespace** and **all necessary header files are included.**
- For each question that asks for the output at a certain line number**,** assume the code has executed correctly up to that point and **give the output produced by that line of code only.** Do not give the output of the entire program. (Note all constructors and destructors have *cout* statements in them).

```
int main(int argc, char* argv[]) {
      MyVect v(10);                          // 1

      v.push(5).push(10).push(15);           // 2

      cout << "v: " << v << endl;            // 3

      MyVect v2 = v;                         // 4
      v2[0] = 20;
      cout << "v2: " << v2 << endl;          // 5

      cout << "v: " << v << endl;            // 6

      v = v;                                 // 7

      cout << "v: " << v << endl;            // 8
      return 0;
}
```

A. (3 points) List the 3 basic copy control operations. Beside each operation, list the line number in *myvect.h* where the operation is declared for the MyVect class.

B. (1 point) The copy constructor of the MyVect class makes a _____ copy. Circle one:

Shallow Deep

C. (1 point) The copy assignment operator makes a _____ copy. Circle one:

Shallow Deep

D. (1 point) What is output when the line marked 1 (in a comment) executes?

E. (1 point) What is output when the line marked 3 executes?

F. (1 point) Why is it possible to chain calls to the push() function together in the line marked with a 2?

G. (1 point) What is output when the line marked 4 executes?

H. (1 point) What is output when the line marked 5 executes?

I. (1 point) What is output when the line marked 6 executes?

J. (1 point) What is output when line 7 executes?

K. (1 point) What is output when line 8 executes?

L. (2 points) What is output when the main function returns?

M. (3 points) This program contains a memory lifetime error(memory leak, double deletion, dangling pointer, etc.). First, identify and describe the error. Second, describe how to fix the error.

Problem 4. (10 points) Using the code provided (ContainerStats.h, ContainerStats.cpp, myvectwithstats.h, myvectwithstats.cpp) and the main function below, answer the following questions.

- Assume we are **using the std namespace** and **all necessary header files are included.**
- For each question that asks for the output at a certain line number**,** assume the code has executed correctly up to that point and **give the output produced by that line of code only.** Do not give the output of the entire program. (Note all constructors and destructors have *cout* statements in them).

```
int main(int argc, char* argv[]) {

     // ContainerStats s(100,0,0);                          // 1

     MyVectWithStats mvs(10, 100, 0, 0);                    // 2

     mvs.insert_and_update(10).insert_and_update(15);       // 3
     ContainerStats* p = &mvs;
     p->print_range();                                      // 4

     p->print();                                            // 5
     return 0;
}
```

A. (1 point) If uncommented, the line marked with a 1 would cause a compilation error. Describe the error?

B. (2 points) What is output when line 2 executes?

C. (1 point) What is the static type of the object pointed to by p?

D. (1 point) What is the dynamic type of the object pointed to by p?

E. (1 point) What is output when line 4 executes?

F. (2 points) What is output when line 5 executes?

G. (2 points) What is output when the main function returns?

Problem 5. (10 points) Using the code provided (ContainerStats.h, ContainerStats.cpp, myvectwithstats.h, myvectwithstats.cpp) and the main function below, answer the following questions.

- Assume we are **using the std namespace** and **all necessary header files are included.**
- For each question that asks for the output at a certain line number**,** assume the code has executed correctly up to that point and **give the output produced by that line of code only.** Do not give the output of the entire program. (Note all constructors and destructors have *cout* statements in them).

```
int main(int argc, char * argv[])
{
      set<MyVectWithStats> s;
      MyVectWithStats V1(10, 100, 0, 0);
      V1.insert_and_update(10);
      V1.insert_and_update(10);
      MyVectWithStats V2(10, 100, 0, 0);
      V2.insert_and_update(50);
      V2.insert_and_update(10);
      MyVectWithStats V3(10, 100, 0, 0);
      V3.insert_and_update(4);
      V3.insert_and_update(15);
      s.insert(V1);
      s.insert(V2);
      s.insert(V3);
      for (auto start = s.begin(); start != s.end(); ++start) {    // 1
            start->print();                                        // 2
      }                                                            // 3
      return 0;
}
```

A. (1 point) By default, what operator does a std::set use to maintain order over the elements it contains?

B. (4 points) True or False:

a. A *set* may contain duplicate copies of a single *key value* _____

b. Associative containers do not provide iterators _____

c. The *key value* in an ordered associative container is *const* _____

d. All sequential containers provide random access iterators _____

C. (5 points) When the for loop in the lines marked 1-3 by comments executes, what is the output produced?

Extra Credit. (2 points) Shared pointers implement reference counting to determine when a heap object should be deleted. For extra credit, you will implement a simple shared pointer struct. Our shared pointer struct will hold ownership of a dynamically allocated Foo object, and safely delete the object when it is no longer pointed to by any shared pointers. To accomplish this, please define the missing functions as described in the comments.

```
struct MySharedPtr {

    Foo * heapObject;
    int * refCount;

    // In this constructor, you should initialize the heapObject member variable
    // with the pointer passed in as the parameter, then dynamically allocate an int
    // and initialize refCount with its address. Set the refCount of the object to 1.
    MySharedPtr(Foo * r)  :
    {









    }

    // The copy constructor should be defined to initialize the 2 member variable pointers
    // as copies of the member variables of the object being copied(a shallow copy). What
    // should happen to  the reference count?
    MySharedPtr(const MySharedPtr & sp)  :
    {









    }
```

```
        // This is the dereference operator, it should be defined to return a
        // reference to the Foo object contained in this shared pointer
        Foo & operator*() {




        }

        // and finally the destructor. The destructor should first decrement the reference count of
        // the heap object pointed to by this shared pointer. Then, if this shared pointer is the
        // last remaining pointer to the dynamically allocated Foo object, it should deallocate the
        // it and the reference count.
        ~MySharedPtr() {









        }
}
```