Name: Kohta Shimbara Pledged: Kohta Shimbara

CSCI 2320 - Data Abstraction Midterm
October 18th, 2021

Turn in procedure:
This midterm is due at midnight. Upload a google doc, pdf, scanned document or any other legible document with your materials. You may take as much time during the day to complete it, but it must be in by midnight or points will be taken off. There are no exceptions to this unless you have come to me beforehand. You should be able to complete this exam in about 45 minutes and you may take class time to do so. If you are unsure about an answer, you may write your reasoning behind your answer and partial credit may be given. All questions assume the C++11 standard.

This exam is open book and open note. You may not share answers or use past exams.

Topic	**Possible Pts**	**Actual**
The C++ Language	38	
Stacks and Queues	16	
Lists and Iterators	28	
File Access	18	
Total	100	

The C++ Language

1. (3 pts) T or F The C++ language is a compiled language.

   T

2. (3 pts) Which of the following statements will compile? (Circle all that apply)
   a. int a = 8.87;
   b. int a2 {8.87};
   c. int a3 = {8.87};

   a,c

For the following two questions, assume the character array contains {'a','b','c','d','e','f','g','h','i','j'} after it is initialized.

3. (6 pts) What will the following code snippet print?

```
char * arr = new char[10];
char c = *((++++arr)+3)
cout << c << endl;
cout << arr << endl;
```

   f
   cdefghij

4. (6 pts) **After running the code in question 3**, which of the following will result in an appropriate delete of the original array 'arr' (no memory error and all memory is deleted)?
   a. delete arr;
   b. delete [] arr;
   c. delete [](----arr);
   d. all of the above
   e. none of the above

   c

5. (5 pts) Suppose a function has the following signature and Node contains a next pointer:
   void foo(Node n, Node * np, Node & nv).
   Which of the following access statements are correct? (Circle all that apply)
   a. Node * next = n.next;
   b. Node * next = np->next;
   c. Node * next = nv.next;

a,b

6. (5 pts) Which of the following is a *pure virtual* function? (Circle all that apply)
    a. virtual int f(int) = 0;
    b. virtual int foo() { cout << "Am I a pure virtual function?\n"; }
    c. void bar() {}
    d. None of the above

    a

7. (5 pts) When sharing mutable data between processes, what c++ structure prevents concurrent access to data?
    a. unique_ptr
    b. shared_ptr
    c. mutex
    d. Thread

    c

8. (5 pts) On a 64-bit machine, which of the following is guaranteed to be at least 64 bits?
    a. sizeof(int)
    b. sizeof(long long)
    c. sizeof(unique_ptr)
    d. sizeof(int *)
    e. None of the above

    b

Stacks and Queues

9. (3 pts) T or F A stack is empty when bottom == 0.

    F

10. (3 pts) How long does it take to insert an item onto a stack?
    a. $O(1)$
    b. $O(n)$
    c. $O(n^2)$
    d. $O(0)$

a

11. (3 pts) T or F The data structure for a stack can be built using a linked list.

T

12. (3 pts) What benefit is there to implementing a queue with a linked list?

Insert and delete will be easier.
It works for variable data. (No need to fix the size at beginning of the implementation)

13. (4 pts) Which of the following are linear data structures? (Circle all that apply)
    a. Stack
    b. Queue
    c. Circular Queue
    d. Array list

a,b,c,d

Lists and Iterators

14. (4 pts) When removing an item from an ArrayList<int>, which of the following operations are necessary in remove(int index)? (Circle all that apply)
    a. for(unsigned int i = index; i<msize-1; ++i) { data[i] = data[i+1]; }
    b. data[index] = 0;
    c. --msize;
    d. delete data[index];

a,c

15. (4 pts) Rank the following iterator types from least (1) to most (4) powerful.
    (1) Output Iterator
    (4) Random Access Iterator
    (2) Forward Iterator
    (3) Bidirectional Iterator

16. (5 pts) Which of the following operations is the fastest? Assume the list is rather large.
    a. Inserting an element into an unsorted doubly linked list at a specific index *i*
    b. Inserting an element into a sorted singly linked list at a specific index *i*
    c. Calling push_back() on an unsorted singly linked list with no tail pointer
    d. Calling pop_front() on an unsorted doubly linked list

    d

17. (5 pts) Suppose a linked list contains the numbers 1 through 10. The erase function takes in an index, removes the element at that index, and updates pointers. What will the list contain after the following code is executed?

```
for (int i = 0; i < 5;++i) {
        lst.erase(i);
}
```

    6 7 8 9 10

18. (4 pts) Using a sentinel node in a linked list implementation removes the need for standalone pointers to:
    a. Head
    b. Tail
    c. Both
    d. Neither

    c

19. (6 pts) What should be placed in the following insert code for a doubly linked list?

```
template<typename T>
typename LinkedList<T>::iterator LinkedList<T>::insert(iterator position,const T &t) {
        Node *rover = position.loc;
        Node *n = new Node(t,rover->prev,rover);
        rover->prev->next  = n;
        rover-> prev = n;
        ++msize;
        return iterator(n);
}
```

File Access

20. (8 pts) What will be in example.txt after running the following code if the file already exists and contains "Hello"?
    What if we change the open mode to "w"?

```
FILE * pFile;
pFile = fopen ( "example.txt" , "a+b" );
fputs ( "First line of output" , pFile );
fseek ( pFile , 11 , SEEK_SET );
fputs ( "Second line" , pFile );
fclose ( pFile );
```

    Hello
    First line of outputSecond line

    First line Second line

21. (10 pts) We have a file-backed linked list with two integers "sz" and "firstFree" that we write at the beginning of our file. Elements have three attributes: T data, int next, int previous. Writes occur by writing data, next, and previous in that order. Suppose we want to read the **data** for the next element of index 4. Write the two seek and two read operations we need to get this data. Perfect syntax is not necessary.

```
fseek(f, 2 * sizeof(int) + (4 * ((2 * sizeof(int) + sizeof(T))) + sizeof(int) + sizeof(T), SEEK_SET);
int next;
fread(&next, sizeof(int), 1, f);
fseek(f, 2 * sizeof(int) + (next * (2 * sizeof(int) + sizeof(T))) + sizeof(int), SEEK_SET);
T data;
fread(&data, sizeof(T), 1, f);
return data;
```