CSCE121: Introduction to Program Design and Concepts

Final Exam

May 3, 2018

Name: _____

UIN: _____

Sections: 505, 506, 507, 508 (Lecture 9:35 a.m.)

Total total pages: 12
Total time: 120 minutes.

Question 1: Summarizing file contents	25
Question 2: Double-ended queue	35
Question 3: Birds of a feather	35
Question 4: Reversing cycles	25
Total	**120**

You are permitted 5 (US Letter) pages of notes, prepared in any way you see fit.

If you're writing in a tight spot and need more space, simply put your answer on another page and add a few words saying where to look for the answer.

Good luck!

Question 1: Summarizing file contents (25)

In this question, you are asked to read in and process a file consisting of multiple lines where each line contains a valid single English word. On the left is an example input file containing such lines; on the right is the desired output for this question (as would be printed to the screen via `cout`):

```
ape
jolly
axe
bicycle
hippopotamus
poetess
angle
trumpet
```

```
Shortest word has 3 letters: ape
Longest word has 12 letters: hippopotamus
The average word length is 6.1250
The mode word length is 6
```

Find the shortest and longest words, and to report them. If there are length ties, print the first instance (as has been done for "ape" above). Furthermore, you need to compute the average and mode word lengths as well. (Recall: the mode is the most frequent among some data; the average is a standard arithmetic mean.)

You are encouraged to use the C++ `string` type to save time, but you may use `char[]`s as well.

Fact: The longest English word, pneumonoultramicroscopicsilicovolcanoconiosis, has 43 letters.

Question 2: Double-ended queue (35)

A FIFO queue allows one to store information so that the first item entered into the queue is also the first to exit from it (FIFO = first-in, first-out). This can be generalized to give a double-ended queue (or *dequeue*, or *deque*). A double-ended queue is a data type that allows you to add an item to the front or the back of the queue, and also to remove an item from either the front or the back.

Write C++ code that defines and implements a double-ended queue data type to store floating point numbers. In addition to creating and destroying a deque, make sure you have functions or methods for the following operations: (1) determine if it is empty; (2) push a number onto the front; (3) pop a number from the front; (4) push a number onto the back; (5) pop a number from the back; (6) read the number at the front; (7) read the number at the back.

You don't know ahead of time how many numbers there will be, but must do this without using any of the C++ standard collections (like `vector`).

Important: Use abbreviations to save time. When two functions are very alike, write one in full detail and then describe what must be changed for the other.

Question 3: Birds of a feather (35)

As a fan of birds and ornithology, you've decided to write some object-oriented C++ code to describe your collection of birds. In this question you make classes to represent birds and to model their properties. You are interested in a particular bird's name, whether that bird is alive or dead, and whether that type of bird can fly. To be most useful, this information should be made available via public methods `getName()`, `isAlive()`, and `isFlightless()`. In addition to these common traits, particular types birds have peculiar aspects (parrots may be able to talk). If particular birds have a peculiar specialized aspect, they should report that via a `getFact()`. Here's a snippet of code making use of the classes:

```
int main(int argc, char **argv) {
    vector<Bird *> birds;
    birds.push_back(new Parrot("Polly", true)); // name + ability to talk
    birds.push_back(new Parrot("Moe", false)); // name + inability to talk
    birds.push_back(new Penguin("Tux")); // name
    birds.push_back(new Owl("Hedwig", 350)); // name + neck rotation range

    birds[1]->passed_on(); // Moe, alas, is no more

    for (Bird *b : birds)
        b->printDetails();

    //... code to clean up, omitted to save space.
}
```

When run, this outputs the following:

```
Polly can fly, is alive. Fact: can talk.
Moe can fly, isn't alive. Fact: can't talk.
Tux can't fly, is alive.
Hedwig can fly, is alive. Fact: neck turns around 350 degrees.
```

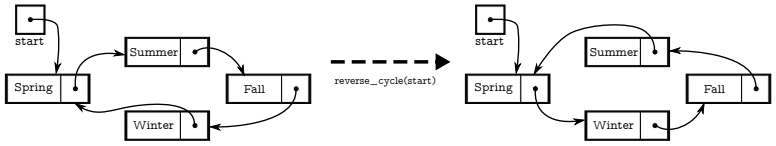
Design and implement the classes to achieve this. Here are some incremental steps to follow:

1. Define a class to represent birds. Infer from the example above what methods are needed.
2. Define `Parrot`, `Penguin`, and `Owl` classes.
3. Complete the code necessary to have `printDetails()` produce the correct output.

Question 4: Reversing cycles (25)

Suppose that you have a cycle of singly-linked structures defined via the following C++ data type:

```
struct cyc_item_t {
    string data;
    cyc_item_t *next;
};
```



Complete the following function so that it reverses the cycle pointed to by its argument. The example in the picture has only four items, but you should ensure that your function works for any number of items.

```
void reverse_cycle(cyc_item_t* &start)
```

END OF THE EXAM