Final Exam Exercises

CS 2308
Spring 2013

Jill Seaman

Chapters 1-7 + 11

Write C++ code to:

- Determine if a number is odd or even
- Determine if a number/character is in a range
  - 1 to 10 (inclusive)
  - between 'a' and 'z' (inclusive)
- Assign a category based on ranges (wind speed)
  - Tropical Depression: <38mph
  - Tropical Storm: 39-73mph
  - Hurricane: >=74mph
  - (extra: there are also 5 levels of hurricanes)

Chapters 1-7 + 11

Write C++ code to:

- Pass arguments by value and reference
  - Return multiple values from a function: compute the area and perimeter of a rectangle
- Process an Array:
  - find maximum/minimum value
  - count values passing a test (>100)
  - sum/average values passing a test (>100)
- Define a structure for weight: lbs and ozs (1lb=16 ozs)
- Determine if one weight is
  - equal to another
  - greater than (or less than) another

Strings

Write C++ function to:

- determine if a string meets the following criteria:
  - it contains only letters, numbers, and the underscore (but none of these is required)
  - the first character is not a digit.
  - You may assume it has at least one character.

Binary Search
Example

The target of your search is 39. Given the following list of integers, record the values of first, last, and middle during a binary search. Assume the following numbers are in an array.

1	7	8	14	20	42	55	67	78	101	112	122	170	179	190
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Repeat the exercise with a target of 179.

Solution for 39 (which is not in the list):

first	0	0	4	4	5
last	14	6	6	4	4
middle	7	3	5	4	x

These are indexes!!

Example Sorting Problem

Given the following list of integers, show what order the integers would be in after executing:

a) one complete pass of the bubble sort

b) two passes of the selection sort.

Assume the following numbers are in an array (subscripts along the bottom).

112	73	8	140	22	42	88	67
0	1	2	3	4	5	6	7

Algorithm Efficiency
Big O Notation

- In order of increasing growth (less efficiency)
  - O(1) constant
  - O(log n) logarithmic
  - O(n) linear
  - O(n log n) linearithmic
  - O(n^2) quadratic
- when using big O notation to describe the efficiency of an algorithm:
  - what is n?
  - what does the function inside the ( ) describe?

Algorithm Efficiency
Give the efficiency of each using big-O notation

- Linear search
- Binary search on an already sorted list
- Bubble sort
- Selection sort
- Access one element in an array
- Array processing:
  - sum, average, show list, find max/min
  - delete all elements

Algorithm Efficiency

Give the efficiency of each using big-O notation

- Linked list operations:
  - insert at head
  - append
  - delete (removeOne)
  - destructor ("delete" all nodes)
  - access one element (by index)
  - sum, average, show list, find max/min (traversal)
  - selection sort, as we did in Assign 6

if you have to traverse the list, its O(n).
if you have to traverse the list once for each element, it's $O(n^2)$

Pointers

- Tracing code with pointers, what is output?

```
int *ptr1, *ptr2;
int foo1, foo2 = 13;

ptr1 = &foo1;
ptr2 = &foo2;
foo1 = 42;
cout << "*ptr1 - " << *ptr1 << endl;
cout << "*ptr2 - " << *ptr2 << endl;

ptr1 = ptr2;
cout << "foo1 - " << foo1 << endl;
cout << "foo2 - " << foo2 << endl;

ptr2 = &foo1;
*ptr1 = *ptr2;
cout << "foo1 - " << foo1 << endl;
cout << "foo2 - " << foo2 << endl;
```


Pointers

- Rewrite using pointer notation instead of []:

```
for (int x = 0; x < 100; x++)
   cout << array[x] << endl;
```

- Write a function to swap the values of its parameters, using pointers instead of pass by reference.
- Write a function to double the size of an array, by dynamically allocating a new array that is twice as big, and copying the elements from the original.
  - who is responsible for deallocating this dynamically allocated array? What statement is used to deallocate it?

Pointers to structs/obj:

- Given the following definitions:

```
struct A {int m, int *x};
A a, *s;
int z;
```

- Write the C++ code for the following:
  - to store the address of a in s.
  - to store 10 in the m field of a (don't use a, use s).
  - to store the address of z in the x field of a.
  - to store 11 in z using only a and x in the statement.
  - to store 12 in z using only s and x in the statement.

Linux Commands

- What linux command would you use to:
  - A. List (display) the files in the current directory?
  - B. Display the name of the current directory?
  - C. Make a new directory called Assignments?
  - D. Make Assignments the current directory?
  - E. Edit a file called myFile.txt?
  - F. Compile a file called myProg.cpp?
  - G. View the contents of myProg.cpp on the screen?
  - H. Delete the file myProg.cpp?
  - I. Execute a makefile?
  - J. Compile a file called a.cpp to an object file?

Classes

- What is the error? (hint: it's a syntax error)

```
class Time {
  private:
   int hour;
   int minute;
  public:
   void addMinute();
   void addHour();
};

void addMinute() {
  if (minute==59) {
    minute=0;
    addHour();
  } else
    minute++;
}
```


Classes

- Given the following Time class:

```
class Time {
  private:
   int hour;
   int minute;
  public:
   Time();
   Time(int, int);
   Time(Time &);
};
```

- Circle the copy constructor prototype
- If t1 is an existing Time object, use the copy constructor to initialize a new Time object t from t1:

Classes

- Given a 24 hour time class, implement display:

```
class Time24 {
  private:
   int hour, minute;
  public:
   Time24(int,int);
   void display();
};
Time24::Time24(int h, int m) { hour = h; minute = m; }
void Time24::display() {

}

int main () { Time24 t(14,23);   t.display();  }
```

Assume the hour will be between 0 and 23 and the minute will be between 0 and 59.

Implement this function to output the time in a 12 hour format with am or pm (but with no 0 padding). The output for the example below should be: 2:23pm

Syntax you should know

- const in member function definition

```
class Time {
    private:
      int hour, minute;
    public:
      int getHour() const;
      Time();
  };
```

const here indicates the function will NOT change any of the member variables' values

- Alternative constructor syntax:

```
Time::Time() : hour(12), minute(0) { }
```

is equivalent to:

```
Time::Time () {
   hour = 12;
   minute = 0;
}
```


Syntax you should know

- setiosflags

```
cout << setiosflags(ios::fixed | ios::showpoint);

cout << fixed << showpoint;
```

These statements are equivalent.

Linked Lists

- Write a function that takes an array of ints and converts it to a linked list by inserting each element to the **front** of the list (the order will be reversed). Return a pointer to the first node.

```
struct Node {
   int data;
   Node *next;
};

Node *convertReverse (int list[], int size) {

      Spoiler alert! Solution on next slide.

}
```


Linked Lists

- Write a function that takes an array of ints and converts it to a linked list by inserting each element to the **front** of the list (the order will be reversed).

```
struct Node {
   int data;
   Node *next;
};

Node *convertReverse (int list[], int size) {
   Node *head = NULL;
   for (int i=0; i<size; i++) {
      Node *newNode = new Node;
      newNode->data = list[i];
      newNode->next = head;
      head = newNode;
   }
   return head;
}
```

Stacks and Queues

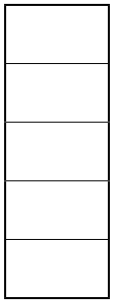
Describe in English how to use a stack to:

- determine if the brackets are matched in a string (or a file).
- evaluate expressions written in post-fix notation.

Stacks and Queues
exercises

Suppose the following operations are performed on an empty stack. Insert numbers in the diagram to show what will be stored in the stack after the operations have executed (label the top):

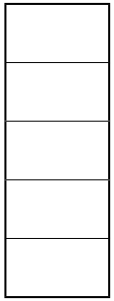
```
int x;
push(3);
push(5);
push(9);
pop(x);
push(2);
pop(x);
push(0);
```



Stacks and Queues
exercises

Suppose the following operations are performed on an empty queue. Insert numbers in the diagram to show what will be stored in the queue after the operations have executed (label the front+rear):

```
int x;
enqueue(3);
enqueue(5);
enqueue(9);
dequeue(x);
enqueue(2);
dequeue(x);
enqueue(0);
```



Specify which location is the front, and which location is the rear. Don't just put the labels at the top or bottom of the drawing.

Disclaimer

- The exercises in this lecture do not cover ALL of the material that will be on the final exam.
- These exercises do provide some sample exam questions.
- There will be questions that will require writing programs or functions or class declarations and implementations.
- There will be questions (short answer, multiple choice) that will test your understanding of the concepts we have covered (vocabulary, etc).