Final Exam Exercises

CS 2308
Fall 2016

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

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
  - sum/average/count values passing a test (>100)
- Define a structure for weight: lbs and ozs (1lb=16 ozs)

Finding Errors

- Input validation for password, 2 errors:

```
PasswordManager pwm;
string pw;
bool testa;
do{
    cout << "Please enter a new password." << endl;
    cin>> pw;
    bool testa = pwm.setNewPassword (pw);
} while(testa = false);
```

- Calculate average with decimals, 2 errors:

```
double average (int a, int b, int c) {
   double result = a + b + c / 3;
   return result;
}
```

Finding Errors

- Sum all values between 0 and 100, 3 errors:

```
int sum (int a[], int size) {
   int total;
   for (int i=1; i<size; i++) {
      if (a[i] >=0 || a[i] <=100)
         total = total + a[i];
   }
   return total;
}
```


Binary Search
Example

The target of your search is 39. Given the following array of integers (values on top, indexes on bottom), record the **values** that 39 is compared to in each iteration of a binary search.

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):

```
67 14 42 20
```


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 does n represent?
  - 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 (with no tail pointer)
  - delete a node with a given value
  - destructor ("delete" all nodes)
  - access one element (by index)
  - sum, average, show list, find max/min (traversal)
  - the sort we did in PA5

if you have to traverse the list, it's 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

- Give the class declaration, and function implementations:

Consider a Student class, which stores a student using three values: a name (string) an ID number (int) and a grade (double).
It has 2 constructors (one default, one with 3 arguments), and set and get function for the name and grade.
The default student has an empty string for the name, and 0 for the other two variables. The other constructor takes the initial name, ID number, and grade.
There is a function print() that outputs the three values.
The student also has a function letter() that calculates and returns the letter grade corresponding to the grade ('A' if grade is 90 or more, 'B' if grade is 80 or more (but less than 90), etc).

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).

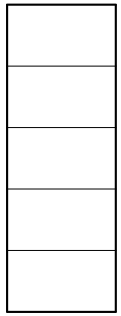
```
struct Node {
   int data;
   Node *next;
};

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
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):

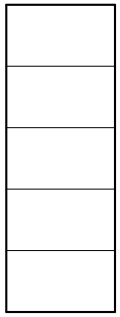
```
int x;
push(3);
push(5);
push(9);
x = pop();
push(2);
x = pop();
push(0);
```



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);
x = dequeue();
enqueue(2);
x = dequeue();
enqueue(0);
```



Stacks and Queues
programming

Implement push, pop, isEmpty and isFull for both of these (and similarly for a Queue):

```
class Stack {
  private:
    static const int
size = 100;
    int stack[size];
    int top;
public:
    Stack() {top = -1;}
    void push (int);
    int pop();
    bool isEmpty();
    bool isFull();
};
```

```
class Stack {
  private:
    struct Node {
      int value;
      Node *next;
    }
    Node *top

public:
    Stack() {top = NULL;}
    void push (int);
    int pop();
    bool isEmpty();
    bool isFull();
};
```


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 (though many will be in multiple choice format on the exam).
- There will be questions that will require writing programs or functions or class declarations and implementations.
- There will be questions (mostly multiple choice) that will test your understanding of the concepts we have covered (vocabulary, etc).