

# Final Exam Review Exercises

CS 2308  
Fall 2011

Jill Seaman

1

## Chapters 1-7

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)
- Determine if one structure/object (Movies) is
  - equal to another
  - greater than (or less than) another
- Assign a category based on ranges (BMI)

2

## Chapters 1-7

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 ( $>100$ )

3

## Ch. 8: Searching and Sorting

Describe each of the following algorithms in English:

- Linear Search
- Binary Search
- Bubble Sort
- Selection Sort

4

## Example Searching Problem

The target of your search is 42. Given the following list of integers, fill in the following table with **the values that you would compare to the target** while executing a:

a) linear search

b) binary search (write the value, not the index to the array). Assume the following numbers are in an array.

1 7 8 14 20 42 55 67 78 101 112 122 170 179 190

--

Repeat the exercise with a target of 82 5

## Example Sorting Problem

Given the following list of integers, show what order the integers would be in after executing one complete, single pass of a:

a) bubble sort

b) selection sort.

Assume the following numbers are in an array.

112 73 8 140 22 42 88 67 9 190

# 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

9

# 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;
```

10

# Pointers

- Rewrite the loop using pointer notation:

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

- Write a function to dynamically allocate a new array of integers of a given size.
  - who is responsible for deallocating this dynamically allocated array? What statement is used to deallocate it?

11

# 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?

12

# 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++;
}
```

13

# 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:

14

# Classes

- What is output?

```
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() {
    string ampm = "am";
    if (hour > 12) {
        int hour = hour-12;
        ampm = "pm";
    }
    cout << hour << ":" << minute << ampm << endl;
}

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

15

## 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;
}
```

16



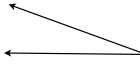
## Syntax you should know

- `setiosflags`

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

```
cout << fixed << showpoint;
```

These statements are equivalent.



17

## 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) {
```

```
}
```

18

## 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;
}
```

19

## 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.

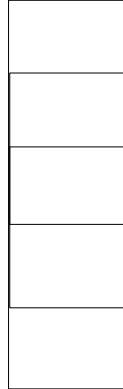
20

## 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);
```



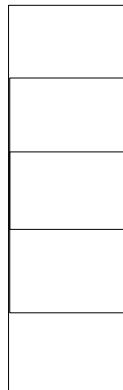
21

## 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);
```



22

# 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).