PA1 – C++ Fundamentals

Due Date

- See Piazza for any changes to due date and time
  - Friday by midnight
  - Grading the next day Saturday Morning
- Submit program to perforce in your student directory
  - Sub directory called:
    - /PA1/...
  - Fill out your ***PA1 Submission Report.pdf***
    - Place it in the same directory as your solution
    - Enter the final Changelist number of your submission
    - Enter the number of test passed
    - Write up a quick discussion in the report
      - What you learned from this assignment

Goals

- Learn
  - C++ basics
    - Classes, methods, pointers, references, scoping
  - Object oriented basics
    - Inheritance, Linked Lists, memory links

Assignments

1. ***Write a several classes to simulate a Chicago Hot Dog Stand.***
   a. create required classes:
      i. HotDog
      ii. Order
      iii. Stand (hot dog stand)
   b. use supplied enums:
      i. Names
      ii. Condiments
   c. You can create additional classes or methods
      i. Especially for debugging and code cleanness

2. ***HotDog*** class - a single hot dog with specific condiments
    a. Use the supplied enumeration class Condiments

```
enum class Condiments
{
      Plain          = 0x0,
      Ketchup        = 0x01,
      Yellow_Mustard = 0x02,
      Green_Relish   = 0x04,
      Chopped_Onions = 0x8,
      Tomato_Wedge   = 0x10,
      Pickle_Spear   = 0x20,
      Sport_Peppers  = 0x40,
      Celery_Salt    = 0x80,
      Everything     = 0xFE
};
```

    b. Create a HotDog or remove the condiments
        i. Add(Condiments ...) one at a time
        ii. Minus(Condiments ...) one at a time
    c. **Everything** is all condiments except for ***Ketchup***
        i. https://en.wikipedia.org/wiki/Chicago-style_hot_dog
        ii. You can add ***Ketchup*** that individually but not part of ***Everything*** option
    d. If you create a HotDog
        i. Its default as ***Plain*** for the condiment list
3. ***Order*** class - contain linked list of specific HotDogs
        i. Use a double linked list to manage orders
            1. With ***next*** and ***prev*** links
        ii. Create an Order then add or remove HotDogs to order
            1. Add(...) - add HotDogs
            2. Remove(...) - remove HotDogs
        iii. Orders are associated to users Name
            1. Use the supplied enumeration class Names

```
enum class Name
{
      Jon,
      Samwell,
      Arya,
      Sansa,
      Tyrion,
      Jaime,
      Cersei
};
```

4. ***Stand*** (hot dog stand) class - holds and manages orders
    i. Use a double linked list to manage orders
        1. With ***next*** and ***prev*** links
    ii. Create a Stand then add or remove Orders to the stand
        1. Add(...) - add Orders
        2. Remove(...) - remove Orders
    iii. Keeps track of the current number of orders
    iv. Keeps track of the peak number of orders
5. Functionality of program
    a. The program must be able to execute sample main() program.
        i. With no linking or compiling errors
    b. The output should mimic as close as possible the supplied output
        i. You are graded on how close your prints look like the sample
    c. Create separate files for each class and header
        i. Do NOT do all your work in main.cpp
    d. NO STL allowed {Vector, Lists, Sets, etc...}
        i. No automatic containers or arrays
        ii. You need to do this the old fashion way - ***YOU EARNED IT***
6. Requirements
    a. Warning Level Wall - no warnings or errors (each one is a point off)
    b. Proper C++ classes
        i. BIg Four operators must be defined
        ii. Custom or default
    c. Use Trace::out() for printing functions
    d. Needs to run in Debug Mode and Release Mode
7. No class can LEAK memory
    a. If a class creates an object using new
        i. It is responsible for its deletion
    b. Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
        i. Leaking is ***HORRIBLE***, so you lose points
8. General
    a. Do all your work by yourself
        i. Feel free to talk with others about ideas on Piazza
        ii. You are 100% responsible for all code
        iii. See syllabus about collaboration rules
    b. Submit your work as you go to perforce several times (at least 5)
        i. As soon as you get something working, submit to perforce
        ii. Have reasonable check-in comments
            1. Seriously, I'm checking
    c. Make sure that your program compiles and runs

i. Warning level ALL sometimes that is not possible due to MS headers…
    1. There are corrections around windows headers
ii. NO Warnings or ERRORS
    1. Your code should be squeaky clean.

9. Submit program to perforce in your student directory
    a. Sub directory called: /PA1/…
        1.

Validation

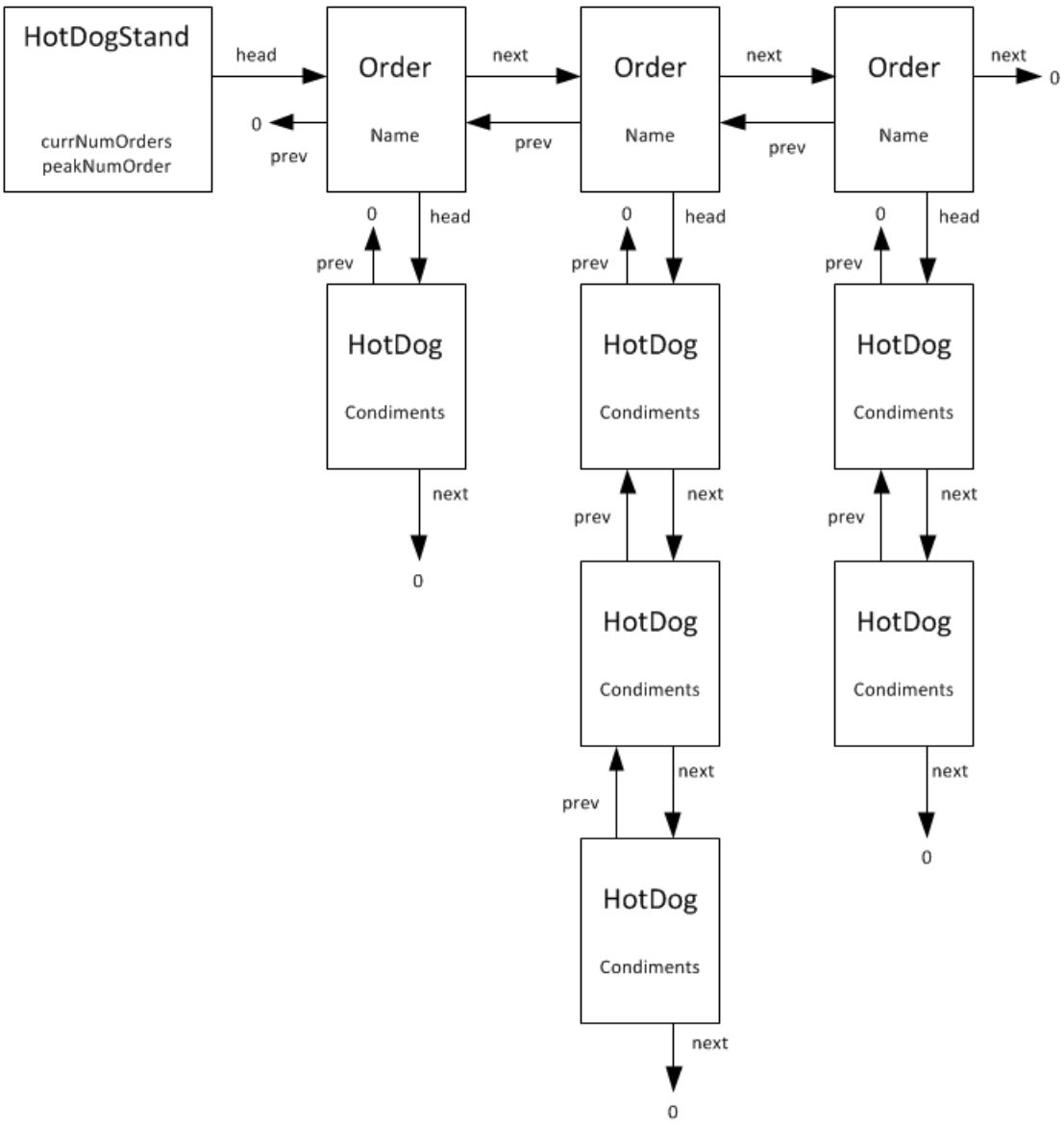
Simple check list to make sure that everything is checked in correctly

- Does your program compile and run without any errors?
- Warning level Wall free?
- Submitted it into /PA1 directory?
- Filled out the submission report?
- Can you delete your local drive, regrab the /PA1 directory?
    - Is all the code there?
    - Does it compile?
    - Your output looks like sample?
    - Leaking memory?

Hints

Most assignments will have hints in a section like this.

- Do many little check-ins
    - Iteration is easy and it helps.
    - Perforce is good at it.
- Look at the lecture
    - A lot of good ideas in there.
    - The code in the examples work.


HotDogStand
currNumOrders
peakNumOrder
head
0
prev
Order
Name
next
Order
Name
next
Order
Name
next
0
prev
prev
0
head
prev
HotDog
Condiments
next
0
HotDog
Condiments
HotDog
Condiments
HotDog
Condiments
HotDog
Condiments
HotDog
Condiments