use Adobe Reader to complete

(Type in fields)

PA1 – C++ Fundamentals

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: [✓ Yes] No

Name:	Joey Domino
Date:	9/17/2021

Submission Details

Final ***Changelist*** number:	113966
Verified build:	[✓ Yes] No
Number Tests Passed:	8
Required Configurations:	DEBUG only

Discussion (What did you learn):

I learned how to:
- Deep copy
- Create double linked lists
- Interface with another class without inheritance
- Create the big 4
- Manipulate Nybles via bitmasks and bitwise operations
- Dynamically allocate and free memory
- Manipulate memory via pointers
- Simulate a hotdog stand ;)

Verify Builds

- Follow the Piazza procedure on submission
  - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
  - No – Generated files
    - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db, *.user
    - Anything that is generated by the compiler should not be included
  - No – Generated directories
    - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
  - *.sln, *.cpp, *.h
  - *.vcxproj, *.vcxproj.filters, CleanMe.bat

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Points will be deducted if minimum is not reached

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn't mean full credit

Submission Report

- Fill out the submission Report
  - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
  - Warning level ALL …
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
  - Code needs to work "as-is".
    - No modifications to files or deleting files necessary to compile or run.
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason…
  - It will not be graded and you get a 0

No Containers

- NO STL allowed {Vector, Lists, Sets, etc...}
    - No automatic containers or arrays
    - You need to do this the old fashion way - ***YOU EARNED IT***

Leave Project Settings

- Do NOT change the project or warning level
    - Any changing of level or suppression of warnings is an integrity issue

Simple C++

- No modern C++
    - No Lambdas, Autos, templates, etc…
    - No Boost
- NO Streams
    - Used fopen, fread, fwrite…
- No code in MACROS
    - Code needs to be in cpp files to see and debug it easy
- ***Exception:***
    - implicit problem needs templates

Leaking Memory

- If the program leaks memory
    - There is a deduction of 20% of grade
- If a class creates an object using new/malloc
    - It is responsible for its deletion
- Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
    - Leaking is ***HORRIBLE***, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
    - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
    - All files must be active to get credit.
    - Better to lose points for unit tests than to disable and lose all points

No Adding files to this project

- This project will work “as-is” do not add files…
- Grading system will overwrite project settings and will ignore any student’s added files and will returned program to the original state

UnitTestConfiguration file (if provided) needs to be set by user

- Grading will be on the UnitTestConfiguration settings
    - Please explicitly set which tests you want graded… no regrading if set incorrectly

Due Dates

- See Piazza for due date and time
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** and commit to perforce
  - ***ONLY*** use Adobe Reader to fill out form, all others will be rejected.
  - Fill out the form and discussion for full credit.

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 : uint8_t
{
        Plain         = 0x0,
        Ketchup       = 0x01,
        Green_Relish  = 0x02,
        Pickle_Spear  = 0x04,
        Tomato_Wedge  = 0x08,
        Yellow_Mustard = 0x10,
        Sport_Peppers = 0x20,
        Celery_Salt   = 0x40,
        Chopped_Onions = 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
      iii. Hint -> Use bitwise manipulations
   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,
        Tyrion,
        Sansa,
        Arya,
        Cersei,
        Jaime,
        Unknown
};
```

1. ***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

2. Functionality of program
   a. The program must be able to Run the unit tests
      i. With no linking or compiling errors
   b. 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***

3. Requirements
   a. Warning Free - no warnings or errors
   b. Proper C++ classes
      i. BIg Four operators must be defined
      ii. Custom or default
   c. Use Trace::out() to help debugging

4. No class can LEAK memory
   a. If a class creates an object using new
      i. It is responsible for its deletion
   b. If a class owns the allocation, it is responsible for deleting it.
      i. Example: Order::Add(HotDog *)
         1. A dynamically created hotdog is passed into this function
         2. When order goes away… so does that hotdog allocation
   c. Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
      i. Leaking is ***HORRIBLE***, so you lose points

Validation

Simple checklist to make sure that everything is submitted correctly

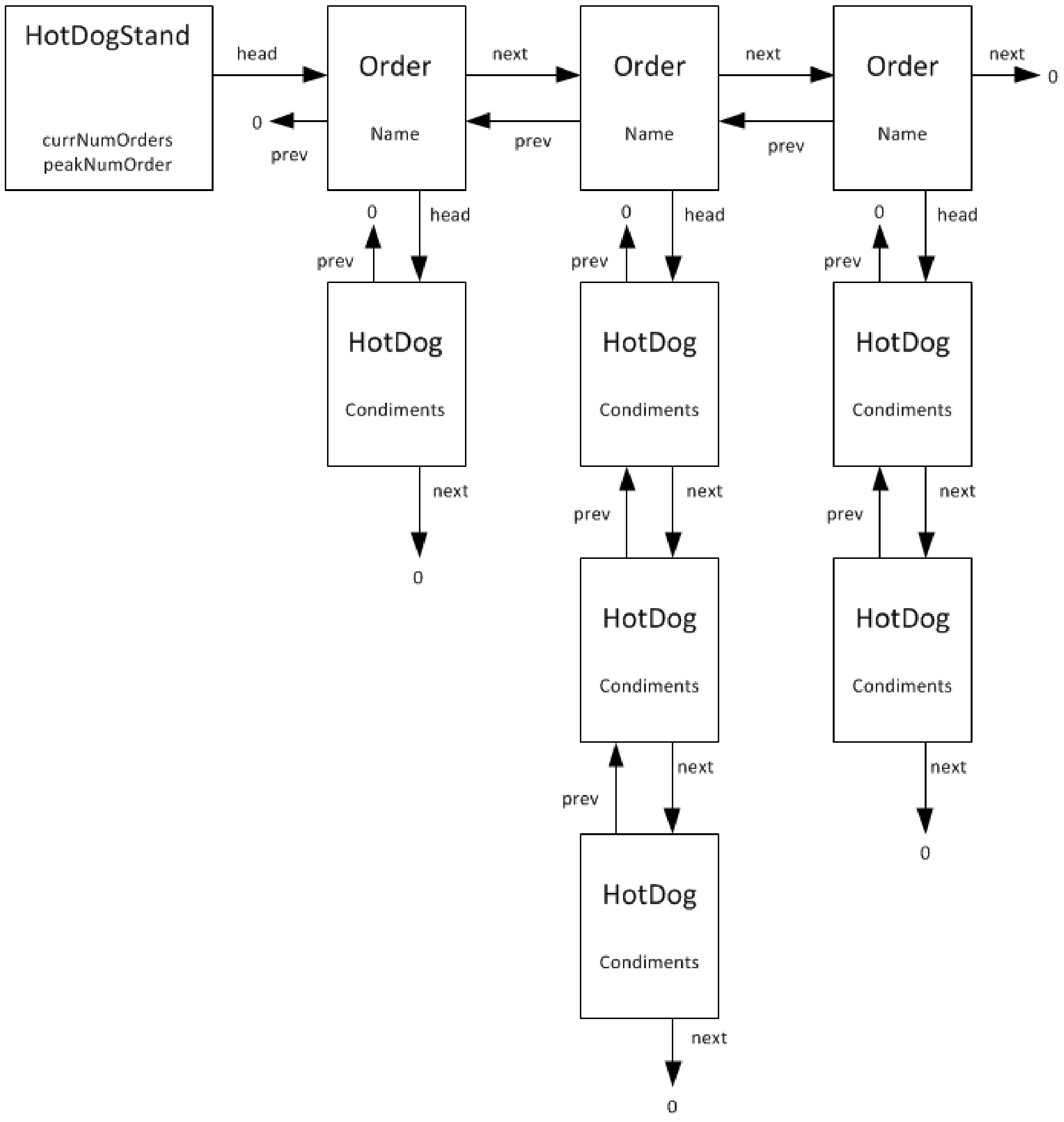
- Is the project compiling and running without any errors or warnings?
- Does the project run ***ALL*** the unit tests execute without crashing?
- Is the submission report filled in and submitted to perforce?
- Follow the verification process for perforce
  - Is all the code there and compiles “as-is”?
  - No extra files
- Is the project 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 tests work – gives hints on how to use the API
- Make several mini projects - will save you time
  - Use your sandbox
    - Experiment with bitwise masking

- Play around with linked lists


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