LAB 11

Merge Sort

LOOK AHEAD

- PA 7
  - Due Monday, April 7th
  - Hope to grade Sunday, April 13th
- Quiz 8
  - Due Monday, April 7th
  - Should be graded by the weekend, April 11th
- PA 8 [Extra Credit]
  - Due Friday, April 11th
  - Unless changed, this is the easy project I mistakenly described as PA 7
  - Hope to grade Sunday, April 13th, possibly the following weekend (April 20th)

LOOK AHEAD (CONT.)

- Quiz 9
  - Due Monday, April 14th
  - Will be graded by Sunday, April 20th
- PA 9 [Group Project 2-4 people]
  - Due Wed, April 23rd
  - *To be graded by Andy*

WILL HOST GITHUB/SFML/NETWORKING TUTORIAL

- GitHub introduction
- Network theory
- SFML setup

SLN PROJECT VS CMAKE PROJECT

SLN:

- Windows-oriented
- Solution manages files
- Largely integrated with Visual Studio
- Used to organize projects
- Possibly better file structure

CMake:

- X-platform
- Integrated with many IDEs
- Specifies build config, source files, libs, and dependencies
- Allows for logic in the build config
- No required file structure

EXAMPLE CMAKE FILE

```
cmake_minimum_required(VERSION 3.10)

project(PA9Project VERSION 1.0)

find_package(SFML 2.5 COMPONENTS graphics window system REQUIRED)

set(CMAKE_CXX_STANDARD 11)

add_executable(PA9 src/main.cpp)

target_link_libraries(PA9 sfml-graphics sfml-window sfml-system)
```

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1		
2	9 5 15 11 7 0 20		
3	9 15 5 20 11 0 7		
4	4 7 2 6 3 1 5		
5	2 3 4 -2 -5 0		

- Is line 1 balanced?

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1	Balanced	
2	9 5 15 11 7 0 20		
3	9 15 5 20 11 0 7		
4	4 7 2 6 3 1 5		
5	2 3 4 -2 -5 0		

- Is line 2 balanced?

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1	Balanced	
2	9 5 15 11 7 0 20	Perfect (Full)	
3	9 15 5 20 11 0 7		
4	4 7 2 6 3 1 5		
5	2 3 4 -2 -5 0		

- Is line 3 balanced?

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1	Balanced	
2	9 5 15 11 7 0 20	Perfect (Full)	
3	9 15 5 20 11 0 7	Perfect (Full)	
4	4 7 2 6 3 1 5		
5	2 3 4 -2 -5 0		

- Is line 4 balanced?

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1	Balanced	
2	9 5 15 11 7 0 20	Perfect (Full)	
3	9 15 5 20 11 0 7	Perfect (Full)	
4	4 7 2 6 3 1 5	Unbalanced	
5	2 3 4 -2 -5 0		

- Is line 5 balanced?

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1	Balanced	
2	9 5 15 11 7 0 20	Perfect (Full)	
3	9 15 5 20 11 0 7	Perfect (Full)	
4	4 7 2 6 3 1 5	Unbalanced	
5	2 3 4 -2 -5 0	Balanced	

- Is line 1 unique?

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1	Balanced	Unique
2	9 5 15 11 7 0 20	Perfect (Full)	
3	9 15 5 20 11 0 7	Perfect (Full)	
4	4 7 2 6 3 1 5	Unbalanced	
5	2 3 4 -2 -5 0	Balanced	

- Is line 2 unique?

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1	Balanced	Unique
2	9 5 15 11 7 0 20	Perfect (Full)	3
3	9 15 5 20 11 0 7	Perfect (Full)	2
4	4 7 2 6 3 1 5	Unbalanced	
5	2 3 4 -2 -5 0	Balanced	

- Is line 4 unique?

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1	Balanced	Unique
2	9 5 15 11 7 0 20	Perfect (Full)	3
3	9 15 5 20 11 0 7	Perfect (Full)	2
4	4 7 2 6 3 1 5	Unbalanced	Unique
5	2 3 4 -2 -5 0	Balanced	

- Is line 5 unique?

BST DIAGRAM PRACTICE

Line #	Input	Balanced?	Other Matches
1	5 2 9 3 1	Balanced	Unique
2	9 5 15 11 7 0 20	Perfect (Full)	3
3	9 15 5 20 11 0 7	Perfect (Full)	2
4	4 7 2 6 3 1 5	Unbalanced	Unique
5	2 3 4 -2 -5 0	Balanced	Unique

MERGE SORT – DIVIDE AND CONQUER

- **Recursion is your friend!**

- Take a list, split it in the middle
  - If it is odd, choose which side has the extra #, maintain the side throughout
- Process each half at a time
- Keep repeating until you have a single element. This is sorted, the base case
- Return merge(left, right)

MERGE SORT – DIVIDE

E FRAME 0 [5, -2, 4, 2, 1, 9]

E – Entry/Exit
R – return
Fn – Frame n

MERGE SORT – DIVIDE

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

E – Entry/Exit
R – return
Fn – Frame n

MERGE SORT – DIVIDE

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

E – Entry/Exit
R – return
Fn – Frame n

MERGE SORT – DIVIDE

E – Entry/Exit
R – return
Fn – Frame n

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

F2 FRAME 3 [5] => a [-2] => b

MERGE SORT – DIVIDE

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

F2 FRAME 3 [5] => a [-2] => b

R FRAME 2 [-2, 5] => a [4] => b

E – Entry/Exit
R – return
Fn – Frame n

MERGE SORT – DIVIDE

E – Entry/Exit
R – return
Fn – Frame n

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

F2 FRAME 3 [5] => a [-2] => b

R FRAME 2 [-2, 5] => a [4] => b

R FRAME 1 [-2, 4, 5] => a [2, 1, 9] => b

MERGE SORT – DIVIDE

E – Entry/Exit
R – return
Fn – Frame n

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

F2 FRAME 3 [5] => a [-2] => b

R FRAME 2 [-2, 5] => a [4] => b

R FRAME 1 [-2, 4, 5] => a [2, 1, 9] => b

F1 FRAME 2 [2, 1] => a [9] => b

MERGE SORT – DIVIDE

E – Entry/Exit
R – return
Fn – Frame n

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

F2 FRAME 3 [5] => a [-2] => b

R FRAME 2 [-2, 5] => a [4] => b

R FRAME 1 [-2, 4, 5] => a [2, 1, 9] => b

F1 FRAME 2 [2, 1] => a [9] => b

F2 FRAME 3 [2] => a [1] => b

MERGE SORT – DIVIDE

E – Entry/Exit
R – return
Fn – Frame n

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

F2 FRAME 3 [5] => a [-2] => b

R FRAME 2 [-2, 5] => a [4] => b

R FRAME 1 [-2, 4, 5] => a [2, 1, 9] => b

F1 FRAME 2 [2, 1] => a [9] => b

F2 FRAME 3 [2] => a [1] => b

R FRAME 2 [1, 2] => a [9] => b

MERGE SORT – DIVIDE

E – Entry/Exit
R – return
Fn – Frame n

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

F2 FRAME 3 [5] => a [-2] => b

R FRAME 2 [-2, 5] => a [4] => b

R FRAME 1 [-2, 4, 5] => a [2, 1, 9] => b

F1 FRAME 2 [2, 1] => a [9] => b

F2 FRAME 3 [2] => a [1] => b

R FRAME 2 [1, 2] => a [9] => b

R FRAME 1 [-2, 4, 5] => a [1, 2, 9] => b

MERGE SORT – DIVIDE

E – Entry/Exit
R – return
Fn – Frame n

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

F2 FRAME 3 [5] => a [-2] => b

R FRAME 2 [-2, 5] => a [4] => b

R FRAME 1 [-2, 4, 5] => a [2, 1, 9] => b

F1 FRAME 2 [2, 1] => a [9] => b

F2 FRAME 3 [2] => a [1] => b

R FRAME 2 [1, 2] => a [9] => b

R FRAME 1 [-2, 4, 5] => a [1, 2, 9] => b

R FRAME 0 [-2, 1, 2, 4, 5, 9]

MERGE SORT – DIVIDE

E – Entry/Exit
R – return
Fn – Frame n

E FRAME 0 [5, -2, 4, 2, 1, 9]

F0 FRAME 1 [5, -2, 4] => a [2, 1, 9] => b

F1 FRAME 2 [5, -2] => a [4] => b

F2 FRAME 3 [5] => a [-2] => b

R FRAME 2 [-2, 5] => a [4] => b

R FRAME 1 [-2, 4, 5] => a [2, 1, 9] => b

F1 FRAME 2 [2, 1] => a [9] => b

F2 FRAME 3 [2] => a [1] => b

R FRAME 2 [1, 2] => a [9] => b

R FRAME 1 [-2, 4, 5] => a [1, 2, 9] => b

R FRAME 0 [-2, 1, 2, 4, 5, 9]

R E [-2, 1, 2, 4, 5, 9]

MERGE SORT - MERGE

- [] [-1 3 5] [1 2 4 6]   ====   [MASTER] [SORTED LEFT] [SORTED RIGHT]
- Start with index 0 on both lists, then merge highest value to master 1 by 1
- [] [**-1** 3 5] [**1** 2 4 6] -1 < 1 => [-1]

MERGE SORT - MERGE

- [] [-1 3 5] [1 2 4 6] ==== [MASTER] [SORTED LEFT] [SORTED RIGHT]
- Start with index 0 on both lists, then merge highest value to master 1 by 1
- [] [**-1** 3 5] [**1** 2 4 6] -1 < 1 => [-1]
- [-1] [~~-1~~ **3** 5] [**1** 2 4 6] 1 < 3 => [-1 1]

MERGE SORT - MERGE

- [] [-1 3 5] [1 2 4 6] ==== [MASTER] [SORTED LEFT] [SORTED RIGHT]
- Start with index 0 on both lists, then merge highest value to master 1 by 1
- [] [**-1** 3 5] [**1** 2 4 6] -1 < 1 => [-1]
- [-1] [~~-1~~ **3** 5] [**1** 2 4 6] 1 < 3 => [-1 1]
- [-1 1] [~~-1~~ **3** 5] [~~1~~ **2** 4 6] 2 < 3 => [-1 1 2]

MERGE SORT - MERGE

- [] [-1 3 5] [1 2 4 6] ==== [MASTER] [SORTED LEFT] [SORTED RIGHT]
- Start with index 0 on both lists, then merge highest value to master 1 by 1
- [] [**-1** 3 5] [**1** 2 4 6] -1 < 1 => [-1]
- [-1] [~~-1~~ **3** 5] [**1** 2 4 6] 1 < 3 => [-1 1]
- [-1 1] [~~-1~~ **3** 5] [~~1~~ **2** 4 6] 2 < 3 => [-1 1 2]
- [-1 1 2] [~~-1~~ **3** 5] [~~1 2~~ **4** 6] 3 < 4 => [-1 1 2 3]

MERGE SORT - MERGE

- [] [-1 3 5] [1 2 4 6] ==== [MASTER] [SORTED LEFT] [SORTED RIGHT]
- Start with index 0 on both lists, then merge highest value to master 1 by 1
- [] [**-1** 3 5] [**1** 2 4 6] -1 < 1 => [-1]
- [-1] [~~-1~~ **3** 5] [**1** 2 4 6] 1 < 3 => [-1 1]
- [-1 1] [~~-1~~ **3** 5] [~~1~~ **2** 4 6] 2 < 3 => [-1 1 2]
- [-1 1 2] [~~-1~~ **3** 5] [~~1 2~~ **4** 6] 3 < 4 => [-1 1 2 3]
- [-1 1 2 3] [~~-1 3~~ **5**] [~~1 2~~ **4** 6] 4 < 5 => [-1 1 2 3 4]

MERGE SORT - MERGE

- [] [-1 3 5] [1 2 4 6] ==== [MASTER] [SORTED LEFT] [SORTED RIGHT]
- Start with index 0 on both lists, then merge highest value to master 1 by 1
- [] [**-1** 3 5] [**1** 2 4 6] -1 < 1 => [-1]
- [-1] [~~-1~~ **3** 5] [**1** 2 4 6] 1 < 3 => [-1 1]
- [-1 1] [~~-1~~ **3** 5] [~~1~~ **2** 4 6] 2 < 3 => [-1 1 2]
- [-1 1 2] [~~-1~~ **3** 5] [~~1 2~~ **4** 6] 3 < 4 => [-1 1 2 3]
- [-1 1 2 3] [~~-1 3~~ **5**] [~~1 2~~ **4** 6] 4 < 5 => [-1 1 2 3 4]
- [-1 1 2 3 4] [~~-1 3~~ **5**] [~~1 2 4~~ **6**] 5 < 6 => [-1 1 2 3 4 5]

MERGE SORT - MERGE

- [] [-1 3 5] [1 2 4 6]   ====   [MASTER] [SORTED LEFT] [SORTED RIGHT]
- Start with index 0 on both lists, then merge highest value to master 1 by 1
- [] [**-1** 3 5] [**1** 2 4 6] -1 < 1 => [-1]
- [-1] [~~-1~~ **3** 5] [**1** 2 4 6] 1 < 3 => [-1 1]
- [-1 1] [~~-1~~ **3** 5] [~~1~~ **2** 4 6] 2 < 3 => [-1 1 2]
- [-1 1 2] [~~-1~~ **3** 5] [~~1 2~~ **4** 6] 3 < 4 => [-1 1 2 3]
- [-1 1 2 3] [~~-1 3~~ **5**] [~~1 2~~ **4** 6] 4 < 5 => [-1 1 2 3 4]
- [-1 1 2 3 4] [~~-1 3~~ **5**] [~~1 2 4~~ **6**] 5 < 6 => [-1 1 2 3 4 5]
- [-1 1 2 3 4 5] [~~-1 3 5~~] [~~1 2 4~~ **6**] 6 => [-1 1 2 3 4 5 6] SORTED!