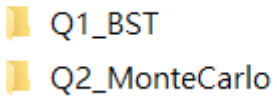
OOP - Midterm 2 – 2022.05.19

Exam Time: 18:30 ~ 21:20
There are four cases for scoring:
If you upload the file:
Before 05/19 21:20, your highest score will be 100.
05/19 21:20 ~ 23:59, your score will reduce 40 points.
05/20 00:00 ~ 21:20, your score will reduce 60 points.
After 05/20 21:20, your score will be 0.
There are two questions for this midterm. Try your best to complete all the questions.

Rules

1. Turn in:
    - Make your directory as below

    - **There should be your entire project (including .sln) in each directory.**
    - Zip all files in one file and name it OOP_mid2_studentID_YourName.zip
    - If you don't follow this rule, your score will reduce 30 points.
    - If you submit wrong code (ex: empty .cpp file), there is no second chance this time and your score is zero.
2. The main.cpp file for each question:
   It is only a basic test for your code. We will also test your code with other test data.
3. During the exam time, the websites you can access are new E3, google meet, Attendance sheet, and Questions.
   You can download your lab codes or assignment codes from new E3 as reference.
   (If new E3 crash down, then no reference if you don't have paper reference.)
   (USB is **NOT** allowed.)
4. You must use the template to do this midterm.
5. You must use visual studio c++ to do this midterm exam.
6. If your source code cannot be built, your score is 0.
7. If you cheat, your score is 0.

Q1 Binary search tree & tree traversal

In this question, there are eleven parts you need to complete.
You need to implement the following functions:

In BST.cpp .

- void Insert(Node* node) : Insert the node to the binary search tree.
- void Inorder_traversal(Node* root) : Use in-order method to traverse binary search tree.
- void Preorder_traversal(Node* root) : Use pre-order method to traverse binary search tree.
- void Postorder_traversal(Node* root) : Use post-order method to traverse binary search tree.
- void Level_traversal(Node* root) : Use level order method to traverse binary search tree.
- int ComputeHeight(Node* root) : Compute the height of the binary search tree.(Assume root level is 1)

In Node.cpp

- int getValue() : Return the value of node.
- Node* getLchild() : Return the left child of this node.
- Node* getRchild() : Return the right child of this node.
- void setLchild(Node* node) : Set the left child of this node.
- void setRchild(Node* node) : Set the right child of this node.

Notice

1. You must complete Node.cpp before you construct the binary search tree
2. The value of each node must be greater than any value stored in the left sub-tree, and less than any value stored in the right subtree.
3. You must use linked list to construct the binary search tree.

Score Distribution(Total: 50 points)

- Inorder traversal: 10 points
- Preorder traversal: 10 points
- Postorder traversal: 10 points
- Level order traversal: 10 points
- Compute Height: 10 points
- All functions in Node.cpp: 0 points
- Insert functions in BST.cpp: 0 points

- If your output format is not the same as we asked, you will not get any points.
- Observe sample output carefully and double check your output result.
- If your program crashes when we test it with our test case, you will not get any points.
- If you can't insert the node to the BST correctly, you will not get any points in this question.

- We will prepare ten test cases to demo your code and compare your output results to our answers. If your output result is not the same as our answer, you will not get any points.

Input Format

The first line shows the number of commands.
The rest are commands.
Command types:

Command	Operation
‘ins’ {value}	Insert a node in the binary search tree
‘ino’	Print the result of the inorder traversal
‘pre’	Print the result of the preorder traversal
‘pos’	Print the result of the postorder traversal
‘lev’	Print the result of the level order traversal
‘hei’	Print the height of the binary search tree

You can assume that the value of each element will not repeat.

Sample Input

// test1.txt
11
ins 9
ins 20
ins 6
ins 17
ins 38
ins 92
ins 41
ins 27
ins 56
ins 3
ino

// test2.txt
13
ins 6
ins 15
ins 13
ins 2
ins 50

ins 81
ins 35
ins 1
ins 14
ins 5
ins 12
ins 37
lev

Sample Output

test1.txt
Inorder_traversal: 3 6 9 17 20 27 38 41 56 92
test2.txt
Levelorder_traversal: 6 2 15 1 5 13 50 12 14 35 81 37

Q2 Monte-Carlo Method

In this question, you need to use Monte-Carlo method to calculate the area of the gray field in each graph.
The side length of each square is 1 unit.

You need to implement the following functions:
In main.cpp:

- void MonteCarlo::generateSamplesFor_CaseOne()
- void MonteCarlo::computeAreaFor_CaseOne()
- void MonteCarlo::generateSamplesFor_CaseTwo()
- void MonteCarlo::computeAreaFor_CaseTwo()
- void MonteCarlo::generateSamplesFor_CaseThree()
- void MonteCarlo::computeAreaFor_CaseThree()
- **void MonteCarlo::quit()**

Notice The number of samples should be 100000

- The answer should be accurate to at least **3** decimal places. (3‰ error allowed, like 0.7824 ~ 0.7884 in case 1 is ok)

Score Distribution(Total: 50 points)

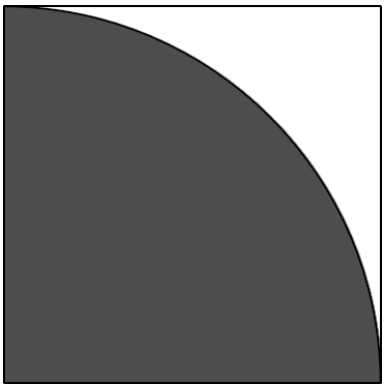
- Graph 1: 25 points
- Graph 2: 15 points
- Graph 3: 10 points
- MonteCarlo::quit() : 0 points

- You **must** use the **Monte-Carlo method** to calculate each answer, or you will not get any points.

- If your output…
  not printing your name and ID before closing
  or print a **wrong answer** (like 0.7**9xx** or 0.7**7xx** in graph 1, because the correct answer is 0.7**8**539)
  or your program **crashes**,
  you will not get any points.

Graph 1 (25 points)

A quarter circle.

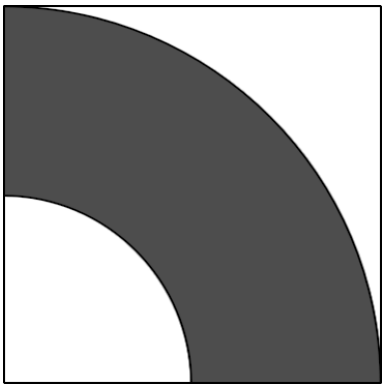


Hint:

1. Define C1 : A quarter circle that origin point is O(0, 0), and radius is 1 unit
2. [In **generateSamplesFor_CaseOne()**]
   Generate lots of random points P(x, y).
3. [In **computeAreaFor_CaseOne()**]
   Calculate distance(O, P), check if **P stays in C1**. $(\overline{OP} \leq 1)$.
   Then calculate the proportion of points within the area and the area of the gray field.

Graph 2 (15 points)

Two quarter circles.

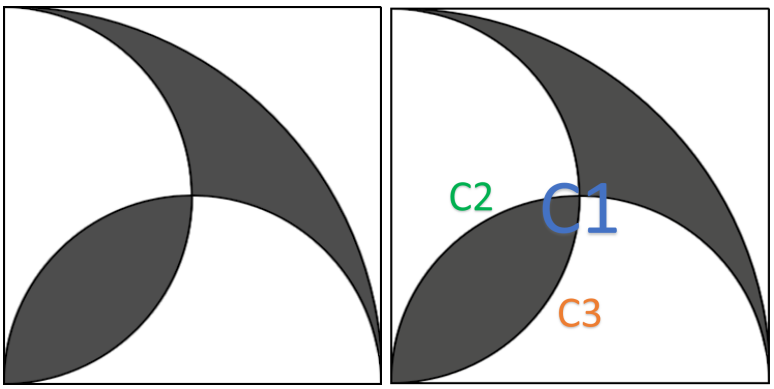


Hint:

1. C1 : A quarter circle that origin point is O(0, 0) and radius is 1 unit
   C2 : A quarter circle that origin point is O2(0, 0) and radius is 0.5 unit
2. Similar to **Graph1**, but change Step 3 to:
   P stays in C1, but not in C2 $(0.5 \leq \overline{OP} \leq 1)$

Graph 3 (10 points)

A quarter circle and two semi-circles.



Hint:

1. C1 : A quarter circle that origin point is O(0, 0) and radius is 1 unit
   C2 : A semi-circle that origin point is O2(0, 0.5) and radius is 0.5 unit
   C3 : A semi-circle that origin point is O3(0.5, 0) and radius is 0.5 unit
2. Similar to **Graph1**, but change Step 3 to:
   (P stays in C1 but not in C2 or C3) or (P stays in both C2 and C3)

Input Format

Just follow the instructions in the program. You don't have to do anything in this part.

Input the case number, then input number of samples (100000).

Output Format

Call showArea(). You don't have to do anything in this part.

Print the shaded area.
It should be accurate to at least **3** decimal places. (3‰ error allowed, like 0.7824 ~ 0.7884 in case 1)

Sample Input

```
1 100000
2 100000
3 100000
0
```

Sample Output (For graph 1)

```
Please enter an option:1
Case:One
Please input the number of samples:100000
Area:0.78435
```

Sample Output (For quit)