Standard Algorithms Quick Revision Using System Design

Name - Pankaj Kumar
Section - KO138
Roll Number - 05
Registration Number - 11910254
Course Name - GEN (331) WORKSHOP ON DESIGN THINKING FOR INNOVATION
Teacher Name - Respected Rameshwar Cambow

CA-3

Introduction

In this project we design a Low-Level System. Which consist most asked interview Question in Tech Company. Simply system consists all Standard Question which are frequently asked in interview. Using this system, we can revise all standard algorithms before our placement. To design the system, we used low level design (LLD).

Low-level design (LLD) is a component-level design process that follows a step-by step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work. Post-build, each component is specified in detail. The LLD phase is the stage where the actual software components are designed. During the detailed phase the logical and functional design is done and the design of application structure is developed during the high-level design phase.

To design the system, we used low level design because low-level design document (LLDD) is to give the internal logical design of the actual program code. Low-level design is created based on the high-level design. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

A good low-level design document makes the program easy to develop when proper analysis is utilized to create a low-level design document. The code can then be developed directly from the low-level design document with minimal debugging and testing. Other advantages include lower cost and easier maintenance.

To revise the algorithms, we have to just execute the code in c++ compiler. Select the question which you want to revise.

For study the algorithms we have to visit the different – different website. But here you can easily excess and revise the algorithms in single place. Like if you selecting any algorithms then you have to choice which things you want about the algorithms like if

you want to sudo code then , you can easily get sudo code , if you want the short description or want to see the working or want to solve the problem based on that algorithms you can easily get.

Using this system, you can quickly revise all the algorithms easily. System has user friendly interface. Like if you are making your project, in your project required any algorithms then you can directly implement these algorithms in your project. If you want modify the system then you modify the system. Simply means you can scale the system. Like if you want to add your own algorithms then you can easily add your algorithms. System architectures are simple. Anyone can understand easily.

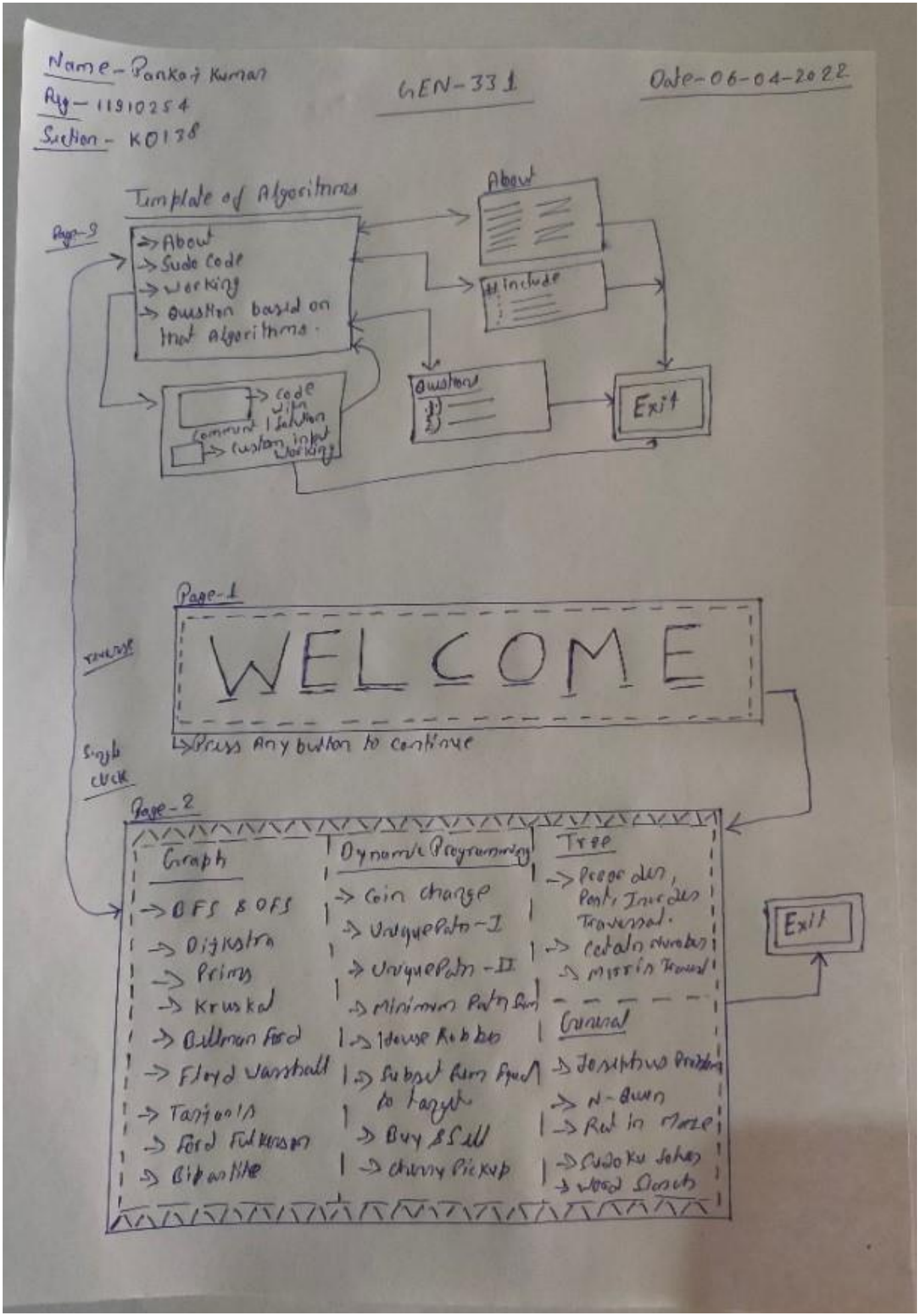
Like If you revise from system then its take very less time compare to hand written Notes. Like if you refer hand written notes then for executing the program you need to write whole program in compiler. But here all programs already running in compiler you need select only algorithms name. after selecting the algorithms, you got everything about algorithms. You can take any number of times custom input (manually taken by user) in this system. You can able see how actually program work from inside. For every algorithm you can get complete details.

For executing the algorithm, you have different options like you execute the program any operating system like – Windows, Mac, Linux. Even you can run any browser like – google chrome, Edge browser, Mozilla Firefox. Even you can also run in Android phone, IOS Phone. Just download C++ compiler or in browser just open any C++ compiler and run the program.

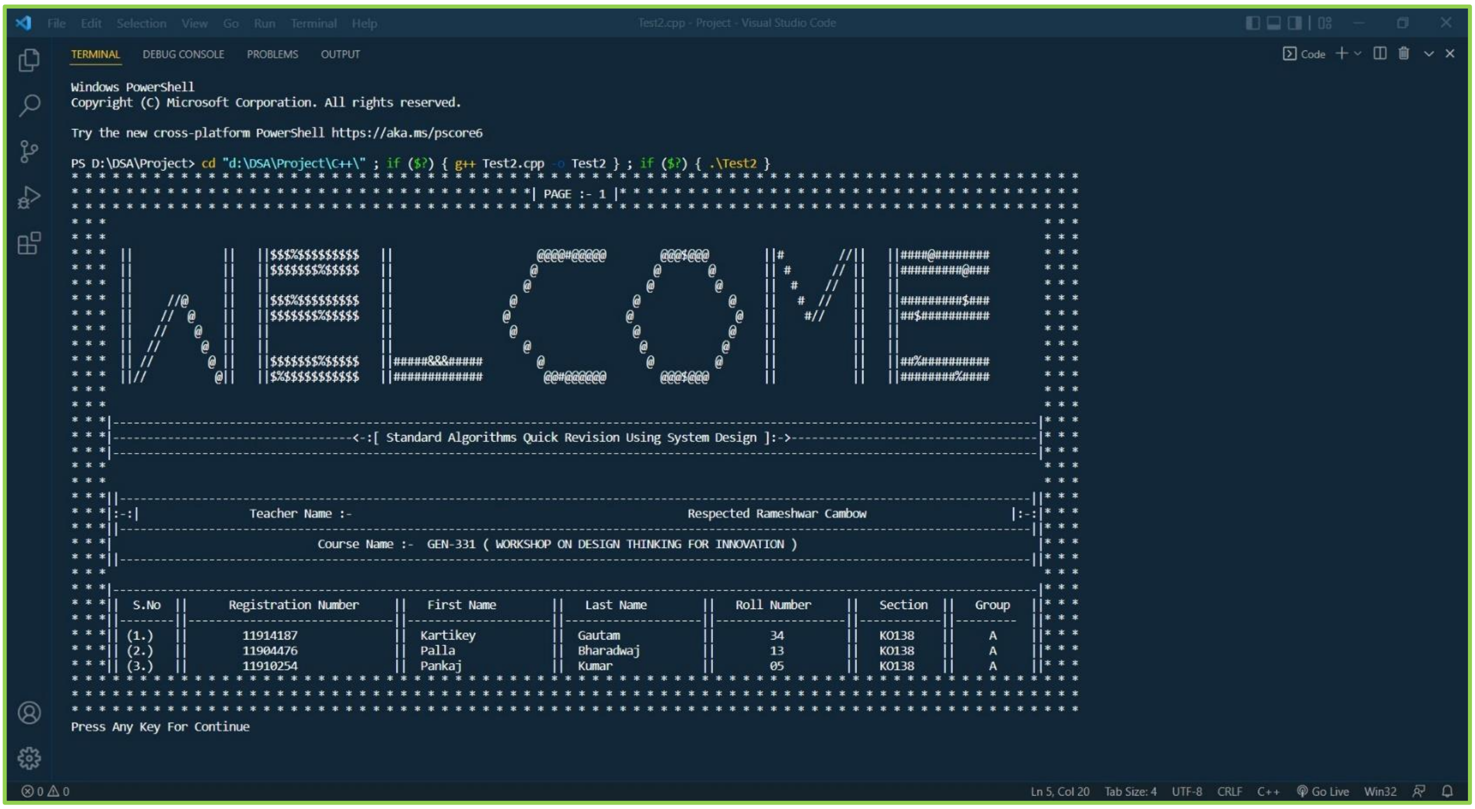
Whole System have less size then you easily share, to your friends. Whole system is optimized, according time complexity and space complexity. That why system take less space and less time to execute. In System every code is written in simple form, even anyone can understand the algorithms easily. System consist 50 standard question, from topic Like Graph, Dynamic Programming, Tree, some general question which are based on math.

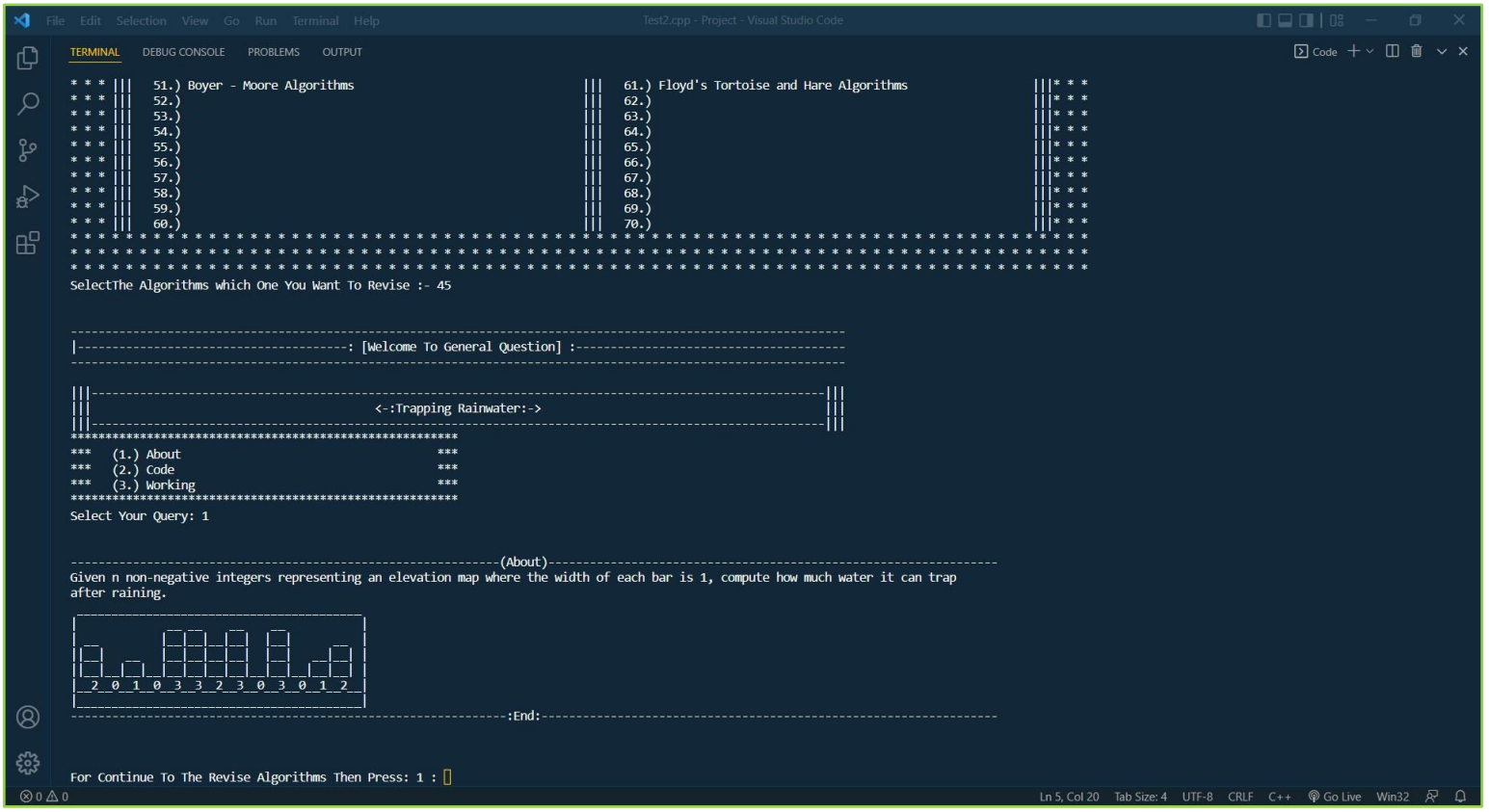
Design and Architectures

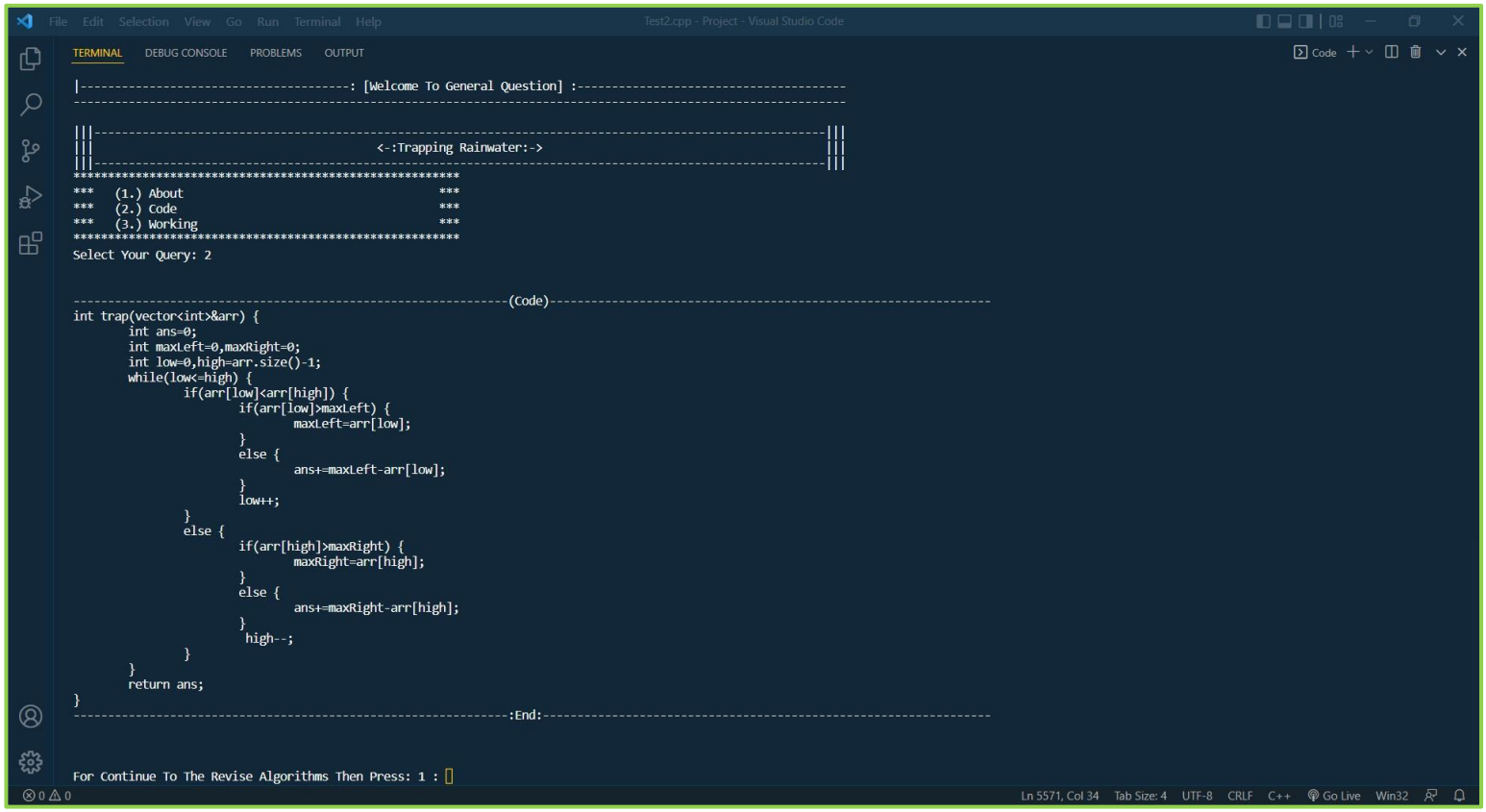
We will design our model on below architectures.

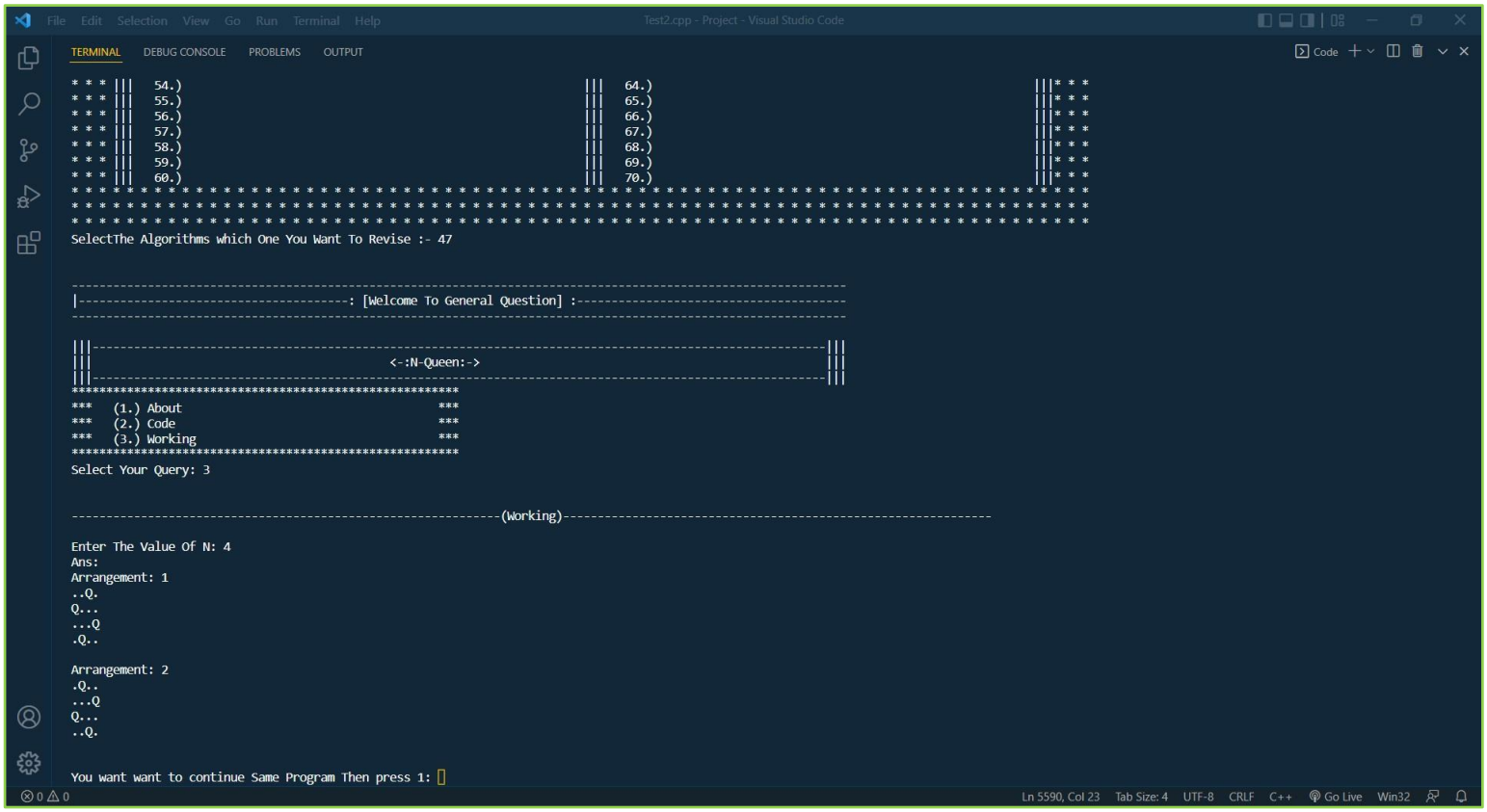


Screenshot of Code




File Edit Selection View Go Run Terminal Help
Test2.cpp - Project - Visual Studio Code
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
Code
* * * ||| 51.) Boyer - Moore Algorithms ||| 61.) Floyd's Tortoise and Hare Algorithms ||| * * *
* * * ||| 52.) ||| 62.) ||| * * *
* * * ||| 53.) ||| 63.) ||| * * *
* * * ||| 54.) ||| 64.) ||| * * *
* * * ||| 55.) ||| 65.) ||| * * *
* * * ||| 56.) ||| 66.) ||| * * *
* * * ||| 57.) ||| 67.) ||| * * *
* * * ||| 58.) ||| 68.) ||| * * *
* * * ||| 59.) ||| 69.) ||| * * *
* * * ||| 60.) ||| 70.) ||| * * *
SelectThe Algorithms which One You Want To Revise :- 45
|----------: [Welcome To General Question] :----------
<-:Trapping Rainwater:->
*** (1.) About ***
*** (2.) Code ***
*** (3.) Working ***
Select Your Query: 1
----------(About)----------
Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.
|_2_0_1_0_3_3_2_3_0_3_0_1_2_|
----------:End:----------
For Continue To The Revise Algorithms Then Press: 1 :
Ln 5, Col 20 Tab Size: 4 UTF-8 CRLF C++ Go Live Win32


File Edit Selection View Go Run Terminal Help
Test2.cpp - Project - Visual Studio Code
TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT
Code
|----------: [Welcome To General Question] :----------
<-:Trapping Rainwater:->
*** (1.) About ***
*** (2.) Code ***
*** (3.) Working ***
Select Your Query: 2
----------(Code)----------
int trap(vector<int>&arr) {
int ans=0;
int maxLeft=0,maxRight=0;
int low=0,high=arr.size()-1;
while(low<=high) {
if(arr[low]<arr[high]) {
if(arr[low]>maxLeft) {
maxLeft=arr[low];
}
else {
ans+=maxLeft-arr[low];
}
low++;
}
else {
if(arr[high]>maxRight) {
maxRight=arr[high];
}
else {
ans+=maxRight-arr[high];
}
high--;
}
}
return ans;
}
----------:End:----------
For Continue To The Revise Algorithms Then Press: 1 :
Ln 5571, Col 34 Tab Size: 4 UTF-8 CRLF C++ Go Live Win32



List of Algorithms

Graph

- ➔ BFS and DFS Traversal
- ➔ Topological Sort
- ➔ Detect Cycle in Directed Graph
- ➔ Detect Cycle in Undirected Graph
- ➔ Shortest Path in DAG
- ➔ Kosarajus Algorithms
- ➔ Dijkstra Algorithms
- ➔ Prims Algorithms
- ➔ Kruskal Algorithms
- ➔ BellmanFord Algorithms
- ➔ FloydWarshall Algorithms
- ➔ isBipartite Graph
- ➔ Articulation Points in Graph
- ➔ Bridge in Graph
- ➔ Tarjan’s Algorithms

➔ Ford Fulkerson Algorithms

Dynamic Programming

➔ Coin Change
➔ Frog Jump
➔ Frog Jump with K^{th} Distance
➔ Buy and Sell
➔ Unique Path Sum - I
➔ Unique Path Sum - II
➔ Minimum Path Sum
➔ Cherry Pickup
➔ Longest Common Subsequence
➔ Longest Increasing Subsequence
➔ Longest Palindromic Subsequence
➔ House Robber
➔ Subset Sum Equal to Target
➔ Partition a Set into two Subsets Absolute Difference
➔ Counts Subset with K Sum
➔ Word Break Problem
➔ Egg Dropping Puzzle
➔ Rod Cutting
➔ Wildcard Matching

Tree

➔ Preorder, Inorder, Postorder Traversal (DFS)
➔ Levelorder Traversal (BFS)
➔ Zig Zag Traversal
➔ Boundary Traversal
➔ Morris Traversal
➔ Lowest Common Ancestor
➔ Catalan Number

General

➔ Josephus Using Recursion
➔ Generate Parentheses
➔ Tower of Hanoi
➔ Trapping Rainwater

➔ Job Sequencing
➔ N-Queen
➔ Rat in a Maze
➔ Sudoku Solver
➔ Word Search
➔ Boyer-Moore
➔ Floyd's Tortoise and Hare (Cycle Detection)

Advantages and Disadvantages

Advantages

->With Click you can execute the whole program.

->You can run on any operating system (any laptop or modern phone).

->You can also directly execute the program, from any web compiler.

->You can easily share the program anywhere. Because program store in single file.

->You can also add your own algorithms. Means easy to scale the system.

->You can able to take your own custom input.

->Easy to use, because user friendly interface.

->For every algorithm you got everything like- about, code, question etc.

->You can also able to modify the code according to you.

->Anyone can understand the code easily.

Disadvantages

->To revise the algorithms, we have to run the program in c++ compiler only.

->If you remove any one line then program not execute.

->You cannot execute the program in another compiler.

->For executing the program, first you have to arrange a laptop or phone.

->Unable to store your previous custom input (given by user manually).

Challenges

When ideas come to my mind about this project. That time I don't know how can design the project and how can convert the project into coding. with help of code how can convert mu real project. That time know little bit of low-level design. Start learning how low-level system architecture's, I can design from many online platforms learn how can design low level system. also start revising my OOPS concept. I design different – different low-level system architectures. After analysis the project I choose that design which can easily understandable to anyone. And easy to scale. When I start implementing my code part that time, I get many errors but I resolve every error make system interface user friendly and make easy to use.

Learning Outcomes

In this project I learn many things like how can convert our idea into the code form. Means how can convert our project into the real project form. which project easy to use to anybody. In this project learn new concept which is OOPs concept. And Low-level system design. My project Source code is more than 7,000 lines. Its is very big project. Learn how can handle the large project in real time. Also learn how can make our project scalable. In this project also learn how can create user friendly interface. In this project I implement different – different standard algorithms which are frequently asked in Tech company. During implementing time, I am able revise my all algorithmic concept. I also learn some new algorithms. Where I do many mistakes and learn how can avoid these types of mistake. In this project I will take survey, from survey I modify my project according to survey response which is given by university student. This project contains more than 50 algorithms, combining all algorithms together. Providing a better interface to student. Where student also can add their favorite algorithms in this project, easily modify the system according to their wish.

Conclusions

Journey start from January where, we daily present our idea to our class teachers. After giving many ideas to sir. Sir finally give the confirmation to my project. Sir told me that you can start your project. My project is “Standard algorithms Quick revision using system design”. In first CA, I take survey about my project collect the repose of student. In second CA, we have to analyze the survey response according to survey we have to modify the project. In third CA we have to implement project into code form where, anyone can execute the program easily and easy to understand. Apart from this from our class teacher – Respected Rameshwar Cambow, who taught many things about project, how can take survey, from survey how can modify the project according our survey. Learn different type of survey. Which survey is good for our project? And learn the process of innovation. How can identify the opportunity areas. How can generate new idea. And learn about the concept of Development and prototype. How can convert our ideas into real project. When I have some doubts about my project then, I always prefer to sir. Who give me right path? Other bundles of things learn from my class teacher. Without sir contribution I unable to make this large project. Finally I am able to complete my project.

References

1.) https://en.wikipedia.org/wiki/Low-level_design

2.)https://www.geeksforgeeks.org/

3.)https://www.interviewbit.com/

4.)https://leetcode.com/

5.)https://cses.fi/