TREASURE HUNTER GAME

Treasure Hunter Game project, it is expected to design a game in which a keyboard-driven character collects treasures on a map with various treasures and obstacles. The aim of the game is for the character to finish the game by collecting all the treasures in the shortest time possible. For this, object oriented programming and data structures knowledge is expected to be used.

Objective: The project implementation aims to reinforce and apply the students' knowledge of object-oriented programming and data structures and to develop problem solving skills.
Programming Language: C++, C#, Java languages can be used in the development of the project.

PROJECT REQUIREMENTS:

In this project, you are expected to help the game character collect all the treasure chests on the grid, starting from a randomly generated starting point on a randomly generated map, without getting stuck in the generated obstacles, in the shortest time and by the shortest path. The character must travel only the necessary paths throughout the game, not the entire grid.

General Steps of the Game:

Generation of the Map, map must be regenerated every time the application starts, and you are expected to develop the algorithm required to generate the map. The algorithm you develop should verify that a random map is generated at each step. You are expected to develop effective algorithms for generation and verification. Effective solutions should be developed as much as possible and brute force applications should be avoided.

When the map is created, the left side of the map should be in winter theme and the right side in summer theme. The images of the fixed objects to be explained in the next section should change according to these themes. Hierarchical object structure should be used to generate fixed objects according to the theme.

You need to make sure that all treasure chests are accessible (not in the middle of obstacles).

Map Content Requirements: The map should contain squares with paths (squares that can be traveled on), obstacles and treasure chests.

Obstacles should be of two types: fixed and movable. Fixed obstacles are objects that do not move during the game, such as trees, mountains, rocks and walls. Separate fixed obstacles should be defined for Summer and Winter. Movable obstacles are objects such as birds and bees, birds can move in an area of 5 squares and bees can move in an area of 3 squares. This movement should be up-down for birds and right-left for bees. Each moving object should occupy an area of 2x2 units.

When each map is created, at least 20 fixed objects (at least 2 of each fixed object must be created) and 3 moving objects must be generated and placed in appropriate places on the map.

Treasure chests should be of four types: gold chest, silver chest, emerald chest and copper chest. Treasure chests are collectible objects. When a character collects a chest, it should disappear on the map. According to the order in which the chests are collected, information should be given in the upper right corner of the screen according to the type of chest (For example, if the collected chest is a gold chest, it should be written "Gold chest collected! Found in (8,16-8,18)").

When each map is created, it is expected that at least 5 pieces of each type of crate are produced and placed in appropriate places on the map.

Determining Start and End Points on the Map: The character's starting point should be randomly assigned to the appropriate (no obstacles or chests) squares on the map. The end point should be the location of the last treasure chest collected according to the developed algorithm.

Expected Outputs: Based on all this information, a map should be created where the character can collect all the treasure chests as soon as possible, starting from a randomized starting point.

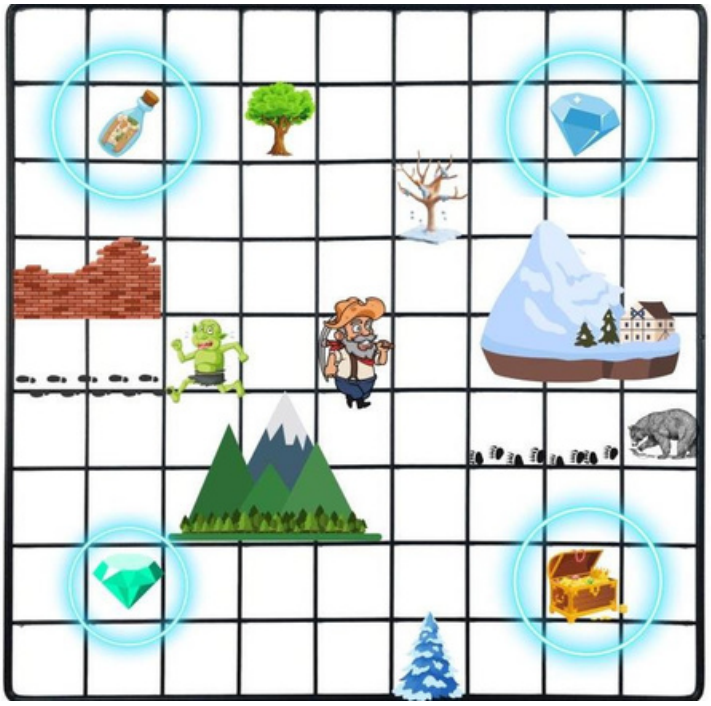

Figure 1. Example grid and obstacle view

Class Definitions:

In the given project, it is expected to create and code classes in accordance with the following definitions.

Character:

Features and functions that should be included in this class:

●ID, Name information should be kept.

●There should be Location variables to hold the coordinates where the characters move.

●Constructor, Get, Set methods must be included.

NOTE: The character cannot move diagonally. It can only move to the right, left, up or down.

Location Class:

●There should be two different variables holding x and y coordinates. Constructor, Get

and Set methods should be included.

Disability Class:

●In the obstacle class, 2 different types of obstacles, fixed and movable, need to be created.

Immobile Barriers:

Trees

●They can be of various types and sizes.
●They can be found in dense woodlands or as solitary trees.
●The player cannot pass over it.
●They can be 2x2, 3x3, 4x4 or 5x5 in size.

Rocks

●They can be of different sizes and shapes (e.g. small pebbles, large rocks).
●The player cannot pass over it.
●They can be 2x2 or 3x3 in size.

Walls

●One wall should not be next to another wall.
●The player cannot pass over it.
●They can be 10x1 in size.

Mountains
●More than one can come together to form mountain clusters.
●The player cannot pass over it.
●They can be 15x15 in size.

Dynamic Barriers:

Birds
●It can only move up and down.
●It can move up and down 5 frames from the square it was created in.
●The player cannot pass over it.
●It should be 2x2 unit size.

Bees
●It can only move in the left-right direction.
●It can move left and right 3 frames from the square where it was created.
●The player cannot pass over it.
●It should be 2x2 unit size.

Application Class:

●Within the application, information such as how many steps the character takes to reach the goal, which objects it obtains, etc. should be kept and displayed on the screen.

NOTE: The above classes are functionally defined in general terms and you are expected to define the properties and methods to be used for each class. In addition, if necessary, you can define different classes from the above classes, provided that you explain why they are used.

In the project, data structures such as tree, queue and Encapsulation, Inheritance,

Polymorphism,The necessary abstraction structures must be used. Every building in the project

Even if you have not used them, you can check whether you know what these structures are

during the project presentation.

Questions will be asked to measure.

Interface and Visuals:

●All visuals in the interface should be clear and unambiguous. The path created, the movement route of moving objects should be clearly visible.
●When the project is run, a randomly generated map design should be displayed. The starting point must be determined.

NOTE: You will receive points for interface design in the given project. Therefore, you are expected to pay attention to the interface. Ready-made designs should not be used for interface design.