Middle East Technical University
Northern Cyprus Campus

CNG242 Programming Language Concepts
Assignment 3 (C++): Gold Rush Alaska Game
Group Project

Date Handed Out: May 18, 2024
Submission Due: May 31, 2024, 23:59

Please Read This Page Carefully

Submission Rules:

- **You are expected to work as a team of students in this assignment. If you cannot find any student to work together, you can work alone.**
- **With your names, surnames, and IDs, you need to write a comment on the first line of your files**, stating that you read the rules specified here and the submission is your own work. **Submissions without this statement will not be graded.** For example,
  /* Javid Babayev – 1234567, Adam Mohammed – 1234568
  We read and accept the submission rules and the extra rules specified in each question. This is our own work that is done by us only */
- **Each team member should write his/her name to the beginning of the method s/he completed. Please note that different grades may be given to the students because of unequal work distribution.**
- Please refer to the syllabus[1] provided for CNG 242 for the measures in place in case of any academic dishonesty[2,3].
- The instructors or TAs may ask for demo sessions for any of the submissions.
- You cannot share this worksheet with any third parties. Upon doing so, any detected action will directly be sent to the disciplinary committee.
- You need to compress your .cpp and .h files and submit a single rar or zip file **named with your student ids** only. Your compressed file should not contain any .exe or any project related 3rd extensions. Only .cpp and .h files will be evaluated. For example, **1234567-1234568.rar or 1234567-1234568.zip.**
- **Only one of the team members should submit the file.**
- Header files should only contain class definitions with prototypes only, no function implementation.
- You should read the questions fully and follow the directions listed in there.
- The assignment should not be shared publicly in any manner, at any time. The assignment cannot be disclosed or disseminated to anyone before, during, or after the submission.
- You cannot use following keywords: allignas, allignof, asm, auto, char8_t, concept, consteval, constexpr, constinit, const_cast, co_await, co_return, co_yield, elifdef, elifndef, export, extern, explicit, goto, import, inline, module, mutable, reflexpr, register, reinterpret_cast, requires, static_assert, synchronized, thread_local, unsigned, volatile, typeid.

---

[1] Page 3&4 (Course rules, #1,2,3)
[2] Taking unfair advantage in assessment is considered a serious offence by the university, which will take action against any student who contravenes the regulation through negligence or deliberate intent.
[3] For a comprehensive cheating definition, please refer to: https://ncc.metu.edu.tr/res/academic-code-of-ethics . When a breach of the code of ethics occurs (cheating, plagiarism, deception, etc.), the student will be added to the BLACKLIST.

Learning Outcomes: On successful completion of this assignment, a student will:

- Combine previously learned concepts to form a project.
- Appreciate clever design and reusability of functionality.

Strategy games typically focus on players' decision-making abilities, often requiring internal decision tree-like thinking and a good sense of situational awareness. On the other hand, board games revolve around pieces or counters moved or positioned on a designated surface, or "board," following a predefined set of rules. While strategic skills are essential for most board games, there is room to incorporate chance elements into their gameplay. Notable examples of strategy board games include Risk, Game of Thrones: The Board Game, Puerto Rico, and Diplomacy.

In this assignment, you will write a program to play a simplified, custom board game called **"Gold Rush Alaska"**. In this game, we will assume that two players (the computer will not play the game) are positioned on opposite ends of a valley, as shown in Figure 1. The valley is a square-shaped platform made of n x n elements, as shown in Figure 2.

Each element on the board may or may not have one type of hidden resource. The hidden elements can be one of the following:

- Food
- Wood
- Medical supplies
- Gold
- Wild animals (the number of wolves should be twice the number of bears)

The players will specify the size of the board for the game. However, the board should always be square, and the size of the board should be at least 5 x 5. For this game, the program should randomly hide $2\lfloor\frac{n^2}{25}\rfloor$ pieces of **<u>each resource</u>** (Food, Wood, Medical, Gold) across the grid (you need to consider an integer division in the given formula). The total number of wolves will be $2\lfloor\frac{n^2}{25}\rfloor$ as well, while the total number of bears is half of this. Please note that $\lfloor\rfloor$ stands for the floor function.

Figure 1: The valley

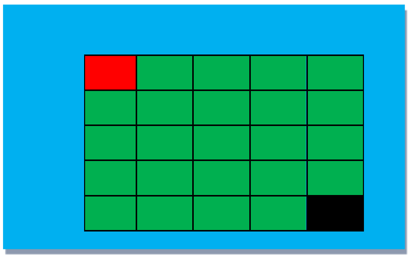


Figure 2: The board

The game begins by generating a random number between 20 and n^2, which indicates the total number of turns (i.e., player 1 and player 2 combined) to take. At each turn, the current state of the game board should be displayed, as well as the total score for each player. Unless specific cell is chosen its contents are unknown to players.

- At the beginning of each turn, the players should choose the [x] and [y] coordinates on the board where the top-left coordinate is (0, 0).
- The rules for placing an item on the game board are as follows:
  - Each player will have a health and score (the amount of Gold found).
  - In the beginning, the health of each player will be **2n,** so if the value of n is 5, then the health of each player is 2*5=10.
  - If a player dies, the other player can continue playing, looking for more Gold.
  - In case there is a wild animal at a specific coordinate, a menu will appear in front of the player to choose an even or odd number. Then, one random number will be generated for the animal (This will be our dice effect). If the player can guess if the random number is even or odd, s/he will manage to escape. Otherwise, the player will receive damage as

follows, and the health of the player will be decreased. If the health becomes 0, then the player will die.

- Damage from wolf $\lfloor \frac{n}{4} \rfloor$.
- Damage from bear $\lfloor \frac{n}{2} \rfloor$.

- If the opened resource is food, wood (for heating and resting), or medical supplies, the health of the player will be increased as follows:
  - Medical supplies $\lfloor \frac{n}{4} \rfloor$
  - Food $\lfloor \frac{n}{6} \rfloor$
  - Wood $\lfloor \frac{n}{8} \rfloor$
- If the opened resource is Gold, the player's score will be increased 100 points, and there will be no change in the player's health.

The objective of the game is to gather as much Gold as possible. The winner is the player with the most resources in terms of points, i.e., the highest score.

On each grid, the resources will be placed randomly and automatically. The effect of the resource is valid independently for each cell. For example, if the cell chosen has a bear which occupies 3 cells, the player can try to guess which other cells are bears to avoid damage, since the other 2 cells will still have the active bear effect. Similarly, if the cell chosen is wood, the player can try to guess the other cell for the effect of the wood. Please note that each element can only be activated once.

The locations and points of each resource can be given as follows:

Element	**Size (number of cells occupied)**	**Character to use for representing it on the board**	**Effect**
Food	1	F	Increase health $\lfloor \frac{n}{6} \rfloor$
Wood	2	I	Increase health $\lfloor \frac{n}{8} \rfloor$
Medical Supplies	1	S	Increase health $\lfloor \frac{n}{4} \rfloor$
Wild animal (Wolf)	1	W	Dice throw, damage $\lfloor \frac{n}{4} \rfloor$.
Wild animal (Bear)	3	B	Dice throw, damage $\lfloor \frac{n}{2} \rfloor$.
Gold	1	G	Increase the score 100 points

Please note that if an element occupies more than one cell, they should all be adjacent cells. Element placement will be discussed in more detail in a later section. After the resources have been positioned, the game proceeds in a series of rounds. In each round, each player takes a turn to announce a target cell in the grid. The computer then announces the actions to be followed. The grid is then updated accordingly.

Implementation

Create a base abstract class, "**Element",** which will have member variables for the size on the board, the character to represent it on the board, and its effect (how much increase or decrease for health). There are five sub-classes for Food, Wood, Medical Supplies, Wild animals, and Gold. Wild animals have two more subclasses, Wolf, and Bear.

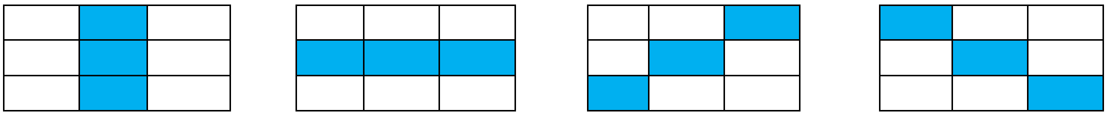
The resources gathered by each player will be held in a structure (you can use any structure, e.g. linked list, array, vector etc.). The structure used to keep the resources will be traversed in each turn to find the total score and current health. **For each three Gold, the player will receive bonus health of $\left\lfloor \frac{n}{4} \right\rfloor$, and for each two woods, will receive bonus health of $\left\lfloor \frac{n}{8} \right\rfloor$ at the end of each round since having Gold,**

and staying warm would positively affect the motivation. For bonus Cumulative amounts will be used as milestones. For example, when 3 gold is reached $\lfloor \frac{n}{4} \rfloor$, health will be added, when 6 is reached another $\lfloor \frac{n}{4} \rfloor$ of health will be added.

A "**Player**" class will be created for each player. Each player is going to have a list of gathered elements logged for the game. A "Grid" class will also be created, which will be associated with the most up-to-date state of the valley. Each player object will have access to the same grid object.

Deployment of Elements

The deploy_elements function of the grid will be called. This function will randomly deploy resources at the beginning of the game. The resources may be placed vertically, horizontally, or diagonally. The 3-cell resources can be placed as shown in the following examples.



Gold hunt

Each player is going to move in turns. Before and after choosing the coordinates, the last version of the grid will be shown. The program should not allow the players to move the same coordinates more than once. So, if a particular coordinate has already been visited, the program will ask for another coordinate.

If all the Gold is gathered, both players die, or the randomly specified number of moves (at the beginning, as mentioned above) is complete, the game will end, and the program will show the winner and loser the details of the resources gathered. In case one of the players dies, s/he can still win the game if the overall score is higher.

Grading Policy

Your submission will be graded as follows:

Item	**Mark (out of 20)**
Player Class	10
Grid Class	25
Elements Class	15
Elements Subclasses	20
Wild Animal Subclass	10
Game Playing	20

Please note that code quality, modularity, efficiency, maintainability, and appropriate comments are part of the grading**.**