Software Technology Department

CCPROG1 Machine Specifications – Pokemon Showdown

Milestone 1: November 16, 2018 **Final Deadline:** December 3, 2018

INTRODUCTION

For your machine project you are to create a basic battle simulator for Pokemon. In the game there should be 9 Pokemon and each one should come from 1 of 3 types: grass, fire, and water. All Pokemon of the same type have the same set of attacks, and each Pokemon has the following stats: HP (hit points) which represent their health, AP(attack points), name and type.

In the game, the user (human player) plays against the computer. Before the battle begins, the program should ask for the user's name and what the user wants to name the player being controlled by the computer. The user will then pick three Pokemon from a list to form his own team while the computer's 3 Pokemon will be randomly picked from the remaining ones. Next, the program should now ask the user which Pokemon they would like to use as their current Pokemon. The program should randomly pick this for the computer as well.

The user will be given the first turn and the turns will be alternating afterwards. For each turn, the user will be given the option to attack, use a potion or switch Pokemon. If the computer is playing one of these will be randomly selected, but it should select attack most of the time (2/3 of the time it should attack, 1/6 of the time it should use a potion, and 1/6 of the time it should switch Pokemon. Also note that grass type attacks are 1.5x more effective against water type Pokemon. Fire type attacks are 1.5x more effective against Grass and Water type attacks are 1.5x more effective against Fire types, A description of each action is given below:

Attack: This gives a list of the different attacks or moves that a Pokemon can make. The user can then select one from this list of moves. Each attack has a name and an attack power and accuracy. The damage caused by each attack should be a random number between the attack power (or the power points if it is lower than the attack power) – 20 and the attack power (or the power points if it is lower than the attack power). The attack should also randomly fail according to the attack's accuracy. The Pokemon types also affect the amount of damage. On the computer's turn just randomly select an attack and use the same formula as above to compute for the damage.

Use Potion: Using a potion gives back 20 HP to the current Pokemon but it consumes the turn (the turn ends after use)

Switch: This should list the stats (Name, Type, HP, AP) of all the user's pokemon. The user can then select from this list and then switch the current Pokemon. If it is the computer who uses this action, then a random Pokemon from the computer's team will be selected for the switch. The turn ends after the switch. **Note:** You cannot switch to a Pokemon with 0 HP.

If either the user or the computer's Pokemon runs out of HP, they should be required to switch Pokemon. If there are no Pokemon left (all have 0 HP), then that player loses.

Continue the game until a player has lost.

GAMEPLAY

1. Ask the user to choose 3 Pokemon. The program should display a list of Pokemon to choose from and the options are shown in the table below.

Type		HP	AP
Grass			
Bulbasaur		160	40
Bellsprout		140	60
Oddish		150	50
Fire			
Charmander		125	70
Ninetails		130	50
Ponyta		140	60
Water			
Squirtle		180	20
Psyduck		170	40
Seel		150	50

2. Ask the user which Pokemon he would like to use first.
3. Generate the computer's team (choose 3 from the remaining Pokemon) and select a starter Pokemon randomly.
4. On each turn the player can either attack, use a potion or switch.
5. At the end of each turn, update the HP of the current Pokemon and check if either side has lost the game.
6. A player has lost if all their Pokemon are out of HP.

Attacks

All Pokemon of the same type have the same attacks

Grass Type		
	Power Points	**Accuracy %**
Leaf Storm	130	90
Mega Drain	50	100
Razor Leaf	55	95

Fire Type		
	Power Points	**Accuracy %**
Ember	60	100
Fire Punch	85	80
Flame Wheel	70	90

Water Type		
	Power Points	**Accuracy %**
Bubble	40	100
Hydro Pump	185	30
Surf	70	90

Damage

The damage of each attack is computed as follows.

Damage = A * Hit * Effectiveness

where

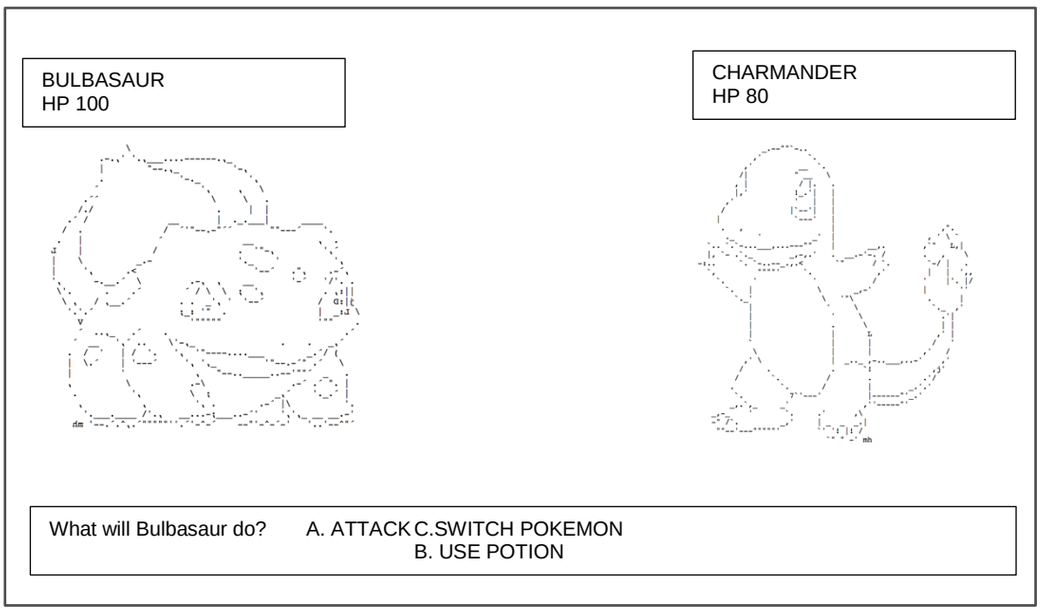
A this should be a random number between the attack power (or the power points if it is lower than the attack power) – 20 and the attack power (or the power points if it is lower than the attack power)

Hit either a 1(hit) or a 0(miss). An attack should miss sometimes based on the accuracy of the attack.

Effectiveness Certain types of Pokemon are more effective than other types. If this is the case the damage becomes 1.5x stronger.

Screen Mockup

Here is a mockup for the battle screens. You are free to add other elements or change the design if you want to as long as the required elements are there.



Milestone 1: Nov 16(F)
Implement the following. This is just the minimum requirement for milestone 1 but you can do always do more ☺

- List of Pokemon
- Selection of Pokemon (User and Computer)
- The user can already choose Attack and Use Potion
- Calculation of damage and updating of HP for the user

Final Deliverable: Dec 3(M)

- Complete game including but not limited to
- Switching between pokemon
- Both the user and the computer must be able to select actions per turn
- Determining when to end the game

Other Instructions

Display the user's current Pokemon along with its stats on the left side of the screen and the computer's current Pokemon and its stats on the right side of the screen.

Make sure that you put appropriate prompts and messages.
Ex. Ash is attacking. Charmander used ember and inflicted 20 damage.

You may use ASCII art for the graphics(Note: You can look at this link https://www.fiikus.net/?pokedex) but if you want to you may add additional features like using graphics libraries, strings and arrays for additional bonus points but make sure to complete all the requirements first. Make sure to consider invalid input, however you don't need to handle cases where the user inputs the incorrect data type (e.g. characters for integer inputs).

You can watch this video of a Pokemon battle as a reference: https://www.youtube.com/watch?v=MYQHFyOjNFY

Bonus Stage: For additional points, add an additional game mode wherein the user can play against a smart computer player that make use of simple AI (artificial intelligence). This should be a separate mode/menu from the original requirements.

Documentation

While coding, you have to include internal documentation in your programs. You are expected to have the following:

- File comments or Introductory comments
- Function comments

File comments are found at the very beginning of your program before the preprocessor directives. Follow the format shown below. Note that items in between < > should be replaced with the proper information and items in between [ ] are considered to be optional parts.

```
/*   Programmed by:    <your name here>   <section>
     Description:      <Describe what this program does briefly>
     Last modified:    <date when last revision was made>
    [Acknowledgements: <list of sites or borrowed libraries and sources>]
*/
<Preprocessor directives>
<function declarations>
main()
{
}
```

Function comments precede the function declaration. These are used to describe what the function does and the intentions of each parameter, if any. Follow the format below when writing function comments:

```
/*   <Description of function>
     [@param  (<type>)  <name>  <purpose>]
     [@return  (<type>) purpose]
*/
<type>
<function name> (<parameter list>)
```

Examples:

```
/*   This function gets the health of the three Pokemon
     @param  (int)  pokemon1_health is health of the first Pokemon,
     @param  (int)  pokemon2_health is health of the second Pokemon,
     @param  (int)  pokemon3_health is health of the third Pokemon,
     @return (void) no return value
*/
int
getHealth( int pokemon1_health,
           int pokemon2_health,
           int pokemon3_health)
{ …
}
```

Other comments in major parts of the code may also be placed in your programs.

Step 3: Testing and Debugging

Submit the list of test cases you have used. For each feature of your program, you have to fully test it before moving to the next feature. Sample questions that you should ask yourself are:

1. What would happen if I input incorrect inputs?
2. Is my program displaying the correct output for all cases?
3. Is my program following the correct sequence of events (correct program flow)?
4. Is my program terminating (ending/exiting) correctly?
5. AND OTHERS…

IMPORTANT POINTS TO REMEMBER:

1. You are required to implement the project using the C language (ANSI C) / Dev C++.

2. The implementation will require you to:
   - Use Functions
   - Appropriately use conditional statements, loops and other constructs discussed in class
   - Follow coding standards

- Include internal documentation

Note: Non-use of self-defined functions will merit a grade of **0** for the **machine project**.

3. For Milestone 1, I will check your code during our class time on the given deadline.
4. For the final deliverable, this should be the final and completed version of your program. All the files for your machine project should be submitted via CANVAS.

   (M) December 3, 2018

   Failure to submit on time will merit a **0.0** for the **MP**.

5. As a backup copy, email your source code to **your my.lasalle account**. Your email should be sent **before** the deadline.

6. You will demonstrate your project on a specified schedule after the submission deadline. Being unable to show up during the demo, or being unable to answer convincingly the questions during the demo will merit a grade of **0.0** for the **course**.

7. Any requirement not fully implemented and instruction not followed will merit deductions.

8. This is an individual project. Working in collaboration, asking other people's help, and/or copying other people's work are considered as cheating. Cheating is punishable by a grade of **0.0** for CCPROG1, aside from which, a cheating case will be filed with the Discipline Office.

This rubric is to be used as a general guideline for evaluating this project.

Rubric for Assessment

Criteria	Exemplary	Satisfactory	Developing	Beginning
Program Correctness	41-70 The application meets all the requirements specified in the project specification. The code is syntactically and logically correctly for all cases. Implementation of the program follows the indicated guidelines and does not violate indicated restrictions. The implementation also exhibits appropriate use of programming constructs.	26-40 The code works for typical input, but fails for minor special cases; the major requirements are met, though some minor ones are not. Some implementation of the program violates indicated restrictions.	11-25 The code sometimes fails for typical input. Many parts of the program implementation violate indicated restrictions and some parts of the solution are not implemented using appropriate programming constructs.	0-10 The code often fails, even for typical input. Most indicated restrictions were violated. *Note: Program that does not run and /or implemented incorrectly (based on specifications and restrictions) automatically gets 0 for this course output.*
Effective Communication / Concept Understanding / Test Cases / Demo	18-20 Answers to questions are correct, reasonable, and reflective of the code. The justifications provided are sound.	11-17 Answers to questions are correct, but some justifications provided are weak.	4-10 Answers to questions are correct, but cannot justify solution (e.g., solution via trial and error, rather than proper understanding and application of concepts).	0-3 Correct understanding of the problem, but was unable to explain workings of code provided. *Note: Failure to explain and justify workings of the code submitted will automatically merit 0 for this course output.*
Readability	9-10 The program conforms to a coding standard that promotes code readability. Internal documentation is comprehensive. *Note: See references for some relevant information on code readability.*	6-8 Minor code formatting does not exhibit consistency in coding standard. Not all functions / program features have proper internal documentation.	1-5 Minimal internal documentation and code readability.	0 No internal documentation and code is not readable.

Note: The maximum score in this rubric for the final MP deliverable is 100 points. An additional 10 points maximum may be given depending on the additional output outlined in the MP specs document. These additional outputs may require advanced reading and understanding of topics that are not covered in the course.