

|  |
| --- |
|  |
| Game Engine Creation  Ass  Ass  2019 |
| December 2022  Student Name: Your Name  Student Number: Your Number |



Contents

[Code must be added as an object, not screenshot 8](#_Toc92189318)

[Section 1: Variables and Operators 8](#_Toc92189319)

[Question 1: User-defined variables (program 4) 8](#_Toc92189320)

[Program Requirements: 8](#_Toc92189321)

[Program Code: 8](#_Toc92189322)

[Program Screenshot: 8](#_Toc92189323)

[Question 2: Area of a Circle (program 8) 9](#_Toc92189324)

[Program Requirements: 9](#_Toc92189325)

[Program Code: 9](#_Toc92189326)

[Program Screenshot: 9](#_Toc92189327)

[Section 2: Conditional statements 10](#_Toc92189328)

[Question 3: Share of Haribo (program 10) 10](#_Toc92189329)

[Program Requirements: 10](#_Toc92189330)

[Program Code: 10](#_Toc92189331)

[Program Screenshot: 10](#_Toc92189332)

[Question 4: Doom Difficulty Menu (program 11) 10](#_Toc92189333)

[Program Requirements: 10](#_Toc92189334)

[Program Code: 11](#_Toc92189335)

[Program Screenshot: 11](#_Toc92189336)

[Section 3: Loops 11](#_Toc92189337)

[Question 5: 10 Numbers (program 14) 11](#_Toc92189338)

[Program Requirements: 11](#_Toc92189339)

[Program Code: 11](#_Toc92189340)

[Program Screenshot: 11](#_Toc92189341)

[Question 6: Starry Output (program 15) 11](#_Toc92189342)

[Program Requirements: 11](#_Toc92189343)

[Program Code: 12](#_Toc92189344)

[Program Screenshot: 12](#_Toc92189345)

[Section 4: Functions 13](#_Toc92189346)

[Question 7: Odds and Evens (program 19) 13](#_Toc92189347)

[Program Requirements: 13](#_Toc92189348)

[Program Code: 13](#_Toc92189349)

[Program Screenshot: 13](#_Toc92189350)

[Section 5: Arrays 14](#_Toc92189351)

[Question 8: Inventory (program 23) 14](#_Toc92189352)

[Program Requirements: 14](#_Toc92189353)

[Program Code: 14](#_Toc92189362)

[Program Screenshot: 14](#_Toc92189363)

[Section 6: Strings 15](#_Toc92189364)

[Question 9: Initials (program 24) 15](#_Toc92189365)

[Program Requirements: 15](#_Toc92189366)

[Program Code: 15](#_Toc92189367)

[Program Screenshot: 15](#_Toc92189368)

[Question 10: String Manipulation (program 27) 15](#_Toc92189369)

[Program Requirements: 15](#_Toc92189370)

[Program Code: 16](#_Toc92189371)

[Program Screenshot: 16](#_Toc92189372)

[Section 7: Debugging 17](#_Toc92189373)

[Question 11: Unexpected Code (program 31) 17](#_Toc92189374)

[Program Requirements: 17](#_Toc92189375)

[Program Code: 17](#_Toc92189376)

[Program Screenshot: 17](#_Toc92189377)

[Section 8: Pointers and References 18](#_Toc92189378)

[Question 12: The Changing Pointer (program 36) 18](#_Toc92189379)

[Program Requirements: 18](#_Toc92189380)

[Program Code: 18](#_Toc92189381)

[Program Screenshot: 18](#_Toc92189382)

[Question 13: Using References (program 38) 19](#_Toc92189383)

[Program Requirements: 19](#_Toc92189384)

[Program Code: 19](#_Toc92189385)

[Program Screenshot: 19](#_Toc92189386)

[Section 9: File Handling 20](#_Toc92189387)

[Question 14: To Ten Scores (program 43) 20](#_Toc92189388)

[Program Requirements: 20](#_Toc92189389)

[Program Code: 20](#_Toc92189390)

[Program Screenshot: 20](#_Toc92189391)

[Section 10: OOP 21](#_Toc92189392)

[Question 15: ePet Care (program 40) 21](#_Toc92189393)

[Program Requirements: 21](#_Toc92189394)

[Program Code: 22](#_Toc92189438)

[Program Screenshot: 22](#_Toc92189439)

[Section 11: GitHub 23](#_Toc92189440)

[Question 16: GitHub Evidence 23](#_Toc92189441)

# Code must be added as an object, not screenshot

# Section 1: Variables and Operators

## Question 1: User-defined variables (program 4)

### Program Requirements:

Write a program, which creates both a **typedef** data type named health of original data type **int** and an enumerated data type named **Weapons**, consisting of the 9 weapons with SWORD starting at the assigned value of 1.

Next create a variable of data type health, with a starting value of 13000, and a variable of data type Weapons, with its initial value to be SAMURAI. Output the results to the console screen.

Output the results to the console screen in the following format:

cout << “My current health is: “ << my\_health << endl;

cout << “And the ID of my weapon of choice is : “ << my\_weapon << endl;

### Program Code:

### Program Screenshot:

## Question 2: Area of a Circle (program 8)

### Program Requirements:

Write a program, which creates a constant variable of float type named **pi**. This will hold the value 3.14159. Next create two variables of floating point data type named **radius** and **areaOfCircle**. Set the starting value of r**adius** to equal 25.0f. Calculate **areaOfCircle**using the following formula: Area of a circle = pi  \* radius \* radius.

Output the result in the following format:

<< “The area of a circle with radius “ << radius << “ is ” << areaOfCircle << endl;

### Program Code:

### Program Screenshot:

# Section 2: Conditional statements

## Question 3: Share of Haribo (program 10)

### Program Requirements:

Pretend that, for some reason, you are now entitled to my stash of Haribo. There are 40 packets available and to make things fair, they are to be shared out equally, as in you all have to have the same number of packets.

On the day when I decide to hand out the sweet sweet Haribo, only 14 students turned up (pretend that a popular video game had been released that day and many students were suddenly too “ill” to turn up).

Write a program that

1. Calculates how many of the 40 packets of Haribo do each of the students get?
2. Calculates how many I get, which will be the remainder left after sharing the 40 packets as equally as possible

Your final answers should be 2 packets for each student, leaving 12 packets left for me. What we want to see is you coding the solution and getting the computer to do the right calculations that come up with these 2 answers. We’re not bothered about the answers themselves, similar to how many mathematics questions are more concerned with “method marks” than the final answer.

### Program Code:

### Program Screenshot:

## Question 4: Doom Difficulty Menu (program 11)

### Program Requirements:

Write a program that displays a short menu such as:

* Please choose your difficulty:
  1. I’m too young to die!
  2. Hey, not to rough!
  3. Hurt me plenty.
  4. Ultra-Violence.
  5. Nightmare…

The program should use a switch statement to display a short message appropriate to the option chosen (such as "Just FYI, they will shoot back").

### Program Code:

### Program Screenshot:

# Section 3: Loops

## Question 5: 10 Numbers (program 14)

### Program Requirements:

Write a program

1. That asks to user to enter 2 numbers.
   1. Using those 2 numbers, calculate the *sum* (total) of the numbers and calculate the *mean* value of those numbers.
   2. Finally, print out the *sum* and *mean* values to the console.
2. Ask the user to enter another number.
   1. Calculate the *sum* and *mean* with the current total and output in the same way as in step 1.
3. Continue to ask the user indefinitely until the user enters a value of zero.

### Program Code:

### Program Screenshot:

## Question 6: Starry Output (program 15)

### Program Requirements:

Write a C++ program

1. Which asks the user for a number *n* between 1 and 10. The program should then print out *n* lines. Each should consist of a number of stars of the same number as the current line number. For example:

Please enter a number: 5

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

1. The user should then be asked if they wish to have another go [Y or N].

* An input of ‘Y’ will
  + Loop the player back to number 1 above,
* An input of ‘N’ will
  + Output a line of text saying “Goodbye.”
  + Pause for a brief second so the user can read the text
  + Then quit.

Hint: you will need to use nested loops!

### Program Code:

### Program Screenshot:

# Section 4: Functions

## Question 7: Odds and Evens (program 19)

### Program Requirements:

Write a program, which asks the user to input 10 integers. These values should be then passed individually to a function, which will determine whether the value is odd or even. The function should return a boolean value.

In the event of the value being odd, it should be added to an odd count, and in the event of it being even it should be added to the even count.

Your program should then call a function which will output how many odd numbers there were and the total, and in a similar manner for the even numbers. Use the following prototype for the output function:

void outputResults ( int numOfOdd, int oddTotal, int numOfEven, int evenTotal );

### Program Code:

### Program Screenshot:

# Section 5: Arrays

## Question 8: Inventory (program 23)

### Program Requirements:

### For this program you will be given a several starting variables and a complete array containing strings. Your job as the programmer is to inform the user, they have picked up a wizard’s staff and ask them if they would like to swap it for the dagger in their inventory.

### The program should have two outcomes:

### The player chooses to keep the staff.

### Update the contents of their inventory accessing the correct element.

### Output the contents of their inventory to screen

### The player chooses to leave the staff behind.

### Inform the player they have left the item behind

### Output the contents of their inventory

### Program Code:

### Program Screenshot:

# Section 6: Strings

## Question 9: Initials (program 24)

### Program Requirements:

Write a program to prompt the user to enter in a single line their first name initial, followed by a space, their middle name initial followed by a space and the entirety of their surname. Store this in a char array.

The program should then output the first initial on one line, then the middle initial on a separate line and the surname on a line of its own.

Note: This program is to demonstrate the use of char arrays. Ensure that you complete this program using a char array and that the data input by the user is stored in a single char array. You can separate out the various parts of the name into separate arrays afterwards, but the initial read from the console should put the entirety of the input into a single char array.

Tips

You will need to loop through your characters to find the first occurrence of a space. This should give you the information you require to be able to access the middle name initial.

### Program Code:

### Program Screenshot:

## Question 10: String Manipulation (program 27)

### Program Requirements:

Write a program to prompt the user to enter their name, which is then stored in a variable of string data type. Insert their name in between the ‘XX’ in the following string:

string testString = “Do you know who loves C++ XX does!”;

Output this string to the console. The program should then remove the two Xs and then output the string to the screen again.

The program should then ask the user to input another name (this one longer than the first). The program should then locate the first name in the test string and replace it with the new name.

Output the altered string with the second name in place of the first to the console and screenshot the entire process.

Note: The second name entered MUST be longer than the first name entered.

Note: This program is to demonstrate the use of the string data type.

Tips

Store the first users name in its own string variable. You can then use this to locate it again when replacing it with the second name.

### Program Code:

### Program Screenshot:

# Section 7: Debugging

## Question 11: Unexpected Code (program 31)

### Program Requirements:

Enter the following program and fix all the functional errors. This program will run, but it will not give the expected results. Copy your fixed code in the relevant slot below along with a screenshot of the working program.

The expected output is:

1 is an odd number

2 is an even number

3 is an odd number

#include <iostream>

using namespace std;

int main()

{

int count = 3;

for( int i = 1; i > count; i++ )

{

cout << i;

if( count = 2 )

cout << “ is an even number” << endl;

else

cout << “ is an odd number” << endl;

}

// Pause the output

cin.get();

return 0;

}

### Program Code:

### Program Screenshot:

# Section 8: Pointers and References

## Question 12: The Changing Pointer (program 36)

### Program Requirements:

Write a program that has integer variables called num1 and num2.

num1 and num2 should have values assigned by request from the user. This must be done in a function called inputDetails().This function should have the following format:

void inputDetails( int\* n1, int\* n2 );

Within the function it should ask the user to input two numbers and then populate the variables appropriately.

Within the main() function create a pointer to an int data type called pNum and point it to num1.

Write another function called outputDetails() which takes num1, num2 and pNum as parameters. This function should output the following details to the console screen:

1. num1 value.
2. num1 address in memory.
3. num2 value.
4. num2 address in memory.
5. pNum value (the address it currently holds)
6. pNum dereferenced value.
7. pNum address in memory.

Ensure the output is referring to num1, num2 and pNum and not local copies. This is where the function prototype is crucial. Your output must be clear. I’d recommend outputting some text to explain each of the above.

Back in the main() function reassign the pointer to point at num2 and output the same as above by calling your outputDetails() function.

Remember to set the pointer to point at nothing once you have finished with it.

### Program Code:

### Program Screenshot:

## Question 13: Using References (program 38)

### Program Requirements:

Write a program that creates an int variable called num. Next create a reference to num called rNum; All the below tasks must be done on the *reference*, with the output of num shown.

1. Ask the user to input a number and store it in num.
2. Output the value of num.
3. Add 25 to the current value stored in num.
4. Output num.
5. Ask the user to input another number and store this in num.
6. Output num.
7. Minus 25 from the current value of num.
8. Output num.

### Program Code:

### Program Screenshot:

# Section 9: File Handling

## Question 14: To Ten Scores (program 43)

### Program Requirements:

This program is to test not only your file handling ability but also how you choose to handle the data internally. Only 10 scores and names should ever be stored. A screenshot of the scores text file should also be included below. It MUST match the output expected from the screenshot of the program running.

Write a program that presents the user with the following option screen:

1. Enter a score
2. Display scores
3. Exit

If the user enters an invalid option the program should inform the user and then re-present the menu screen.

Upon entering ‘1’ the user should be asked for a score and a name. If the entered score is greater than the lowest score already in the file, or there are less than 10 scores stored this new score should be incorporated in to the top ten. This should then be saved in a text file called ‘scores.txt’. The user is then returned to the menus screen.

Upon entering ‘2’ the program should output a well-formatted list of scores. This should stay on screen until the user presses the return key. If there are no scores the user should be informed and then returned to the menu screen.

Upon entering ‘3’ the program should close. All file streams should be closed correctly.

NOTE: This program must not crash. It is up to you as programmer to ensure invalid options are dealt with and that if the file doesn’t exist and the user tries to access it that they are informed, and the program returns to the menu.

### Program Code:

### Program Screenshot:

# Section 10: OOP

## Question 15: ePet Care (program 40)

### Program Requirements:

### You will need four class files, Pet must be the parent and then three child classes, Dog, Cat and Hamster. These child functions need just three methods, constructor and destructor and use of the virtual Talk method you will create in Pet.

### In Pet you will need:

### A constructor which takes two pre-set integer parameters which are set to 0 in the braces (hunger and boredom)

### A destructor

### A virtual void method called Talk

### Two void methods

### Feed and Play, each taking a related a single pre-set int food or fun, both set to 4

### (int food = 4)

### Two private member variables

### m\_hunger and m\_bored (set to 0)

### two protected methods

### An inline constant function that takes type int and returns m\_hunger + m\_bored called GetMood

### And a void PassingTime which takes a pre-set int called time that is equal to 1

### The source file needs to contain the following:

### The Constructor

### Cout that a new pet has arrived

### Set m\_hunger to equal hunger (hunger being the passed variable) and the same with m\_bored equal to boredom

### Feed

### Cout a message that the animal has eaten e.g. “Burp!”

### m\_hunger needs to be set to -= food

### create an if statement that controls if m\_hunger falls below 0 then it is set to 0, preventing negative numbers

### call PassingTime

### Play

### Same as above but with fun and m\_bored

### Talk

### Cout “I am your pet and I feel “

### Create an int mood and set to equal the GetMood method call

### Add an if, else if, else statement

### If mood above 15 cout “mad”

### If above 10 – “frustrated”

### If above 5 – “okay”

### Else “happy”

### Call PassingTime

### PassingTime

### M\_hunger += time and the same for m\_bored.

### The child override Talk method must state which animal it is but otherwise be pretty much the same as the Pet::Talk method. Feel free to change the mood values as you like depending on the animal.

### In the main, create a pointer to Pet as you did with mammal, and ask the user which pet they want. Depending on the choice set the pointer to equal a new animal similar to the mammal program and call Talk(). Once this has been done create a loop with a switch statement. Give the user the following options:

### 0 – Quit

### 1 – Listen to your pet

### 2 – Feed your pet

### 3- Play with the pet

### These cases should call the appropriate method

### Run the program and ensure it works correctly. Screenshot should show the pets mood changing and show when they Talk it states the correct animal too.

### Program Code:

### Program Screenshot:

# Section 11: GitHub

## Question 16: GitHub Evidence

For this section you must provide evidence of your continued use of GitHub throughout the semester. This can be in the form of screenshots of your commits and branches, and a link to your account. This section will be marked on a curved based off the quality of use of GitHub.

* 0 marks – No use of GitHub or account creation
* 2 marks – Account creation only
* 5 marks – Account creation and one commit on main branch for the project
* 6 marks – Account creation with regular commits on the main branch
* 8 marks – Account creation with use proper use of branches for weeks/tutorials and regular commits
* 10 marks – As above with detailed, well formatted README doc