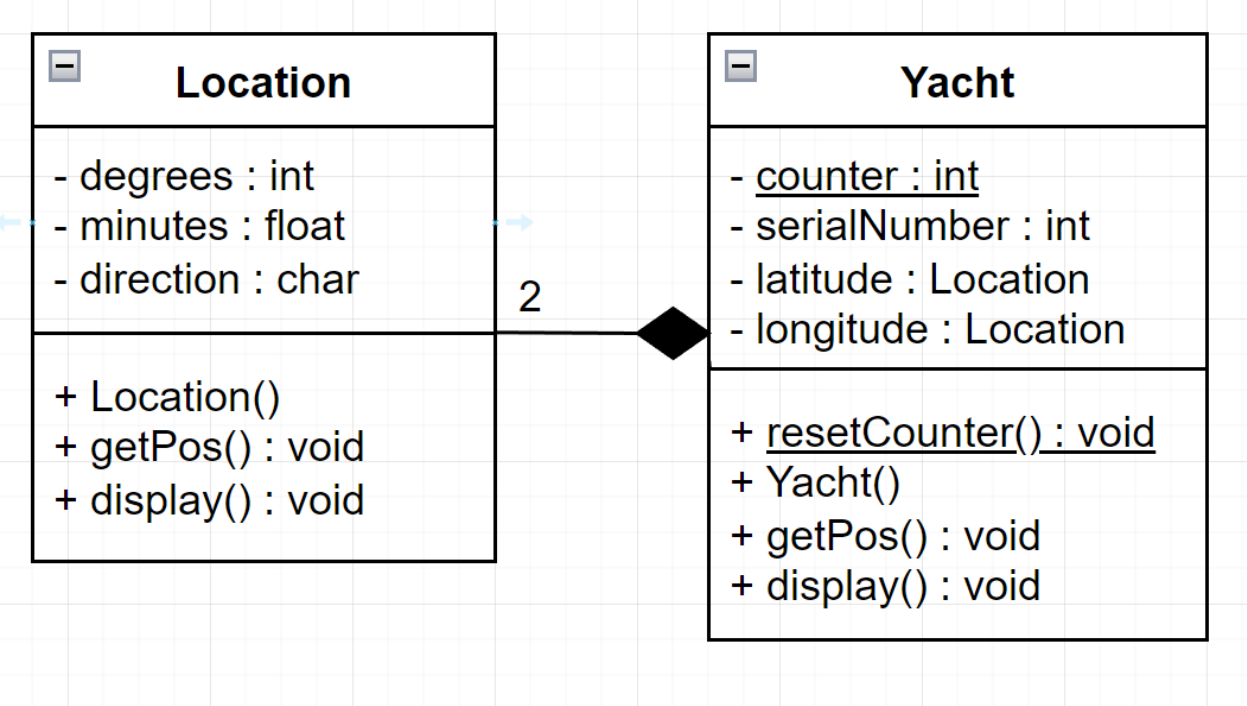
Alexander Legner – 270168960

UML Diagrams

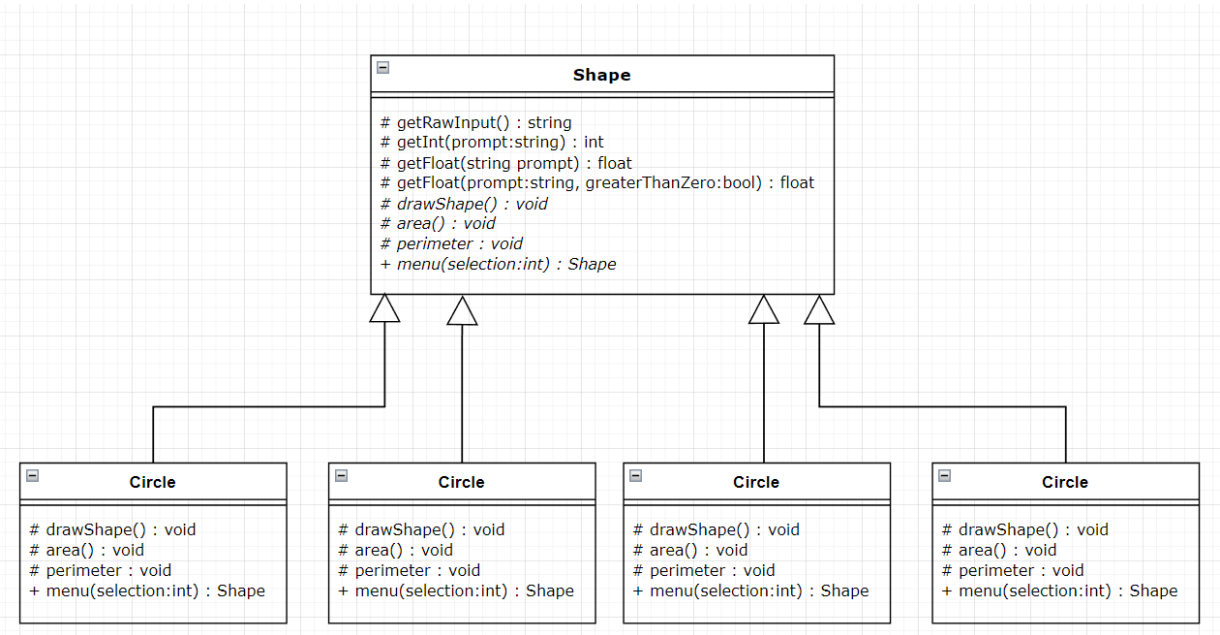
Task 1

Prepare the UML class diagram for the Location class:




Task 3

Draw the class diagram UML clearly depicting the access specifiers, inheritance, data members and member functions of all classes planned to use



Describe The Process – Task 1

To start off, I decided to make a Yacht class as outlined in brief. So, I added the first two functions outlined in the brief: getPos() and display().

```
class Yacht
{

public:
    /// <summary>
    /// Gets latitude and longitude for the yacht
    /// </summary>
    void getPos();

    /// <summary>
    /// Displays the final positition of the yacht
    /// </summary>
    void display();
};
```

When I was going to code these, I realized that they both used functionality that was going to be made in the Location class outlined in the brief. So, I changed my focus and went straight into making a location class to allow my Yacht class functions to be able to be coded.

```
class Location
{
    int degrees;
    float minutes; // Seconds not required as minutes take decimals for seconds
    char direction; // Compass ('N','E','S','W')
public:
    /// ...
    Location();

    /// ...
    void getPos();

    /// <summary>
    /// Display the location latitude and longitude in (###°##' D) format
    /// </summary>
    void display();
};
```

The header file shows that the location class contains all the required members. The degrees, minutes, and direction were all private as they weren't needed outside the Location class. They were going to used to store either a Latitude or Longitude which have the same concept but are used together to accurate define a location which is how naval vessels, like a yacht, can navigate at sea. I decided I should create the functionality for these functions in the Location.cpp file next so I know how they will be used in the Yacht class.

```cpp
void Location::getPos()
{

    bool degreesVaild = false; // Loop condition
    while (!degreesVaild) { // Loop while getting input for degrees
        std::cout << "Input the degrees between 0 and 180: "; // Prompt

        // Raw input string from console
        std::string rawInput;
        std::getline(std::cin >> std::ws, rawInput);

        try {
            // Check for decimals
            int len = rawInput.find_last_of('.'); // Get decimal places - i.e. 1.2 is has 1dp
            // Stops decimal input
            if (len > 0) {
                throw std::invalid_argument("Input cannot have decimals...");
            }

            // Check input is in range
            int testValidInput = std::stoi(rawInput); // Convert input to int value
            if (testValidInput >= 0 && testValidInput <= 180) { // Range check
                degrees = testValidInput; // Sets valid input
                degreesVaild = true; // Stop the loop
            }
            else { // Stop values that exceeds range
                throw std::invalid_argument("Input exceeds boundaries...");
            }
        }
        catch (...) { // Catch all exceptions
            std::cout << "Invalid input...\n"; // Inform user and loop
        }
    }
```

```cpp
#include <string>
#pragma once
/// ...
class Location
{
    int degrees;
    float minutes; // Seconds not required as minutes take decimals for seconds
    char direction; // Compass ('N','E','S','W')

    void getDegrees();
    void getMinutes();
    void getDirection();

public:
    /// ...
    Location();

    /// ...
    void getPos();

    /// <summary>
    /// Display the location latitude and longitude in (###°##' D) format
    /// </summary>
    void display();
};
```

When I started coding getPos, I realized that the function was getting long and hard to make changes. So, I split it up into three private functions which get each part of the Location.

This made the getPos function easy to understand and make edits to each functions algorithm to get each type easier.

```cpp
void Location::getDegrees() { ... }

void Location::getMinutes() { ... }

void Location::getDirection() { ... }

void Location::getPos()
{
    getDegrees();
    getMinutes();
    getDirection();
}
```

```cpp
void Location::getDegrees()
{
    bool degreesVaild = false; // Loop condition
    while (!degreesVaild) { // Loop while getting input for degrees
        std::cout << "Input the degrees between 0 and 180: "; // Prompt

        // Raw input string from console
        std::string rawInput;
        std::getline(std::cin >> std::ws, rawInput);

        try {
            // Check for decimals
            int len = rawInput.find_last_of('.'); // Get decimal places - i.e. 1.2 is has 1dp
            // Stops decimal input
            if (len > 0) {
                throw std::invalid_argument("Input cannot have decimals...");
            }

            // Check input is in range
            int testValidInput = std::stoi(rawInput); // Convert input to int value
            if (testValidInput >= 0 && testValidInput <= 180) { // Range check
                degrees = testValidInput; // Sets valid input
                degreesVaild = true; // Stop the loop
            }
            else { // Stop values that exceeds range
                throw std::invalid_argument("Input exceeds boundaries...");
            }
        }
        catch (...) { // Catch all exceptions
            std::cout << "Invalid input...\n"; // Inform user and loop
        }
    }
}
```

I will only show one of the 3 functions because they are similar, just have a different input type / requirement. These functions use a while loop and a try catch to ensure user input into the console is within an acceptable range, doesn't have decimals if value is an int, and is a valid number if a float or int. This also stops errors occurring if there are conversion errors between string and number. After I completed the user input for getting data in for the location, I made a constructor which set the values into NULL to allocate memory when an instance of Location was created.

```
Location::Location()
{
    // null / empty values to assign memory to members

    degrees = NULL;
    minutes = NULL;
    direction = NULL;
}
```

Afterwards I added to console output in the formatted way required. This then completed the location class, and I was ready to make my way back to creating the Yacht class.

```
void Location::display()
{
    // Outputs to the console in the format (Degrees Minutes Direction)
    std::cout << degrees << '\xF8' << minutes << '\'' << (char)(std::toupper(direction));
}
```

Completed Location class.

```
#include <iostream>
#include "Location.h"

void Location::getDegrees() { ... }

void Location::getMinutes() { ... }

void Location::getDirection() { ... }

void Location::getPos()
{
    getDegrees();
    getMinutes();
    getDirection();
}

void Location::display()
{
    // Outputs to the console in the format (Degrees Minutes Direction)
    std::cout << degrees << '\xF8' << minutes << '\'' << (char)(std::toupper(direction));
}

Location::Location()
{
    // null / empty values to assign memory to members

    degrees = NULL;
    minutes = NULL;
    direction = NULL;
}
```

I came back to the Yacht header and added two location members to the class: Latitude and Longitude. While

```
class Yacht
{
private:
    Location latitude;  // Degrees, Minutes, Direction
    Location longitude; // Degrees, Minutes, Direction
```

I was in the header file, I also added the serial Number of the Yacht and a static counter variable into the class header.

```
class Yacht
{
private:
    static int counter; // Count is used for unique serial numbers
    int serialNumber;   // Unique serial number of instance
```

```
public:
    /// <summary>
    /// Default constructor for a yacht
    /// </summary>
    Yacht();
```

I almost thought I was ready to start coding the functions in the Yacht class but when I was about to start, I realized I needed a constructor for the Yacht class. The reason for this was because I needed a way to assign the serial number and increment the static counter variable.

```
Yacht::Yacht()
{
    serialNumber = counter++; // Set serial number THEN increment counter
    latitude = Location();  // Set blank location in memory
    longitude = Location(); // Set blank location in memory
}
```

The constructor of a yacht creates two Location instances to fill the latitude and longitude members and assign them memory. I also assigned the current count of the number of Yachts to the serialNumber member and THEN incremented the counter. This is done by adding ++ to the end of counter as it will only increment by +1 after assigning the value. One issue I encountered was that I needed the counter to start from 1, and the static variable was not being initialized meaning it had no memory. To fix this I added a reset counter function which assigned the counter to number of my choosing and then assigned the value memory by setting it to NULL in the Yacht.cpp file.

```
int Yacht::counter = NULL; // Assign the counter memory

void Yacht::resetCounter(int resetVal)
{
    Yacht::counter = resetVal;
}
```

I had to make sure the counter was reset to 1 on the start of the program so I added the function into the main function now so that I don't forget.

```
int main()
{
    Yacht::resetCounter(1); // Resets the static int counter
```

```
void Yacht::getPos()
{
    std::cout << "Enter the location of ship #" << serialNumber << ":\n"; // Prompt
    latitude.getPos();  // Gets location values for degrees minutes direction for latitude
    longitude.getPos(); // Gets location values for degrees minutes direction for longitude
}
```

I added in the functionality for getting the latitude and longitude to make it execute the same as shown in the example in the brief.

I also did the same thing for displaying the instance of a Yacht

```
void Yacht::display()
{
    // Displays the formatted yachts position
    std::cout << "The ship serial number is: " << serialNumber << std::endl;
    std::cout << "The position of ship #" << serialNumber << " is ";
    latitude.display(); // Latitude in correct format
    std::cout << " Latitude\t";
    longitude.display(); // Longitude in correct format
    std::cout << " Longitude\n";
}
```

Completed Yacht class.

```
#include <iostream>
#include "Yacht.h"
#include "Location.h"

int Yacht::counter = NULL; // Assign the counter memory

void Yacht::resetCounter(int resetVal)
{
    Yacht::counter = resetVal;
}

Yacht::Yacht()
{
    serialNumber = counter++; // Set serial number THEN increment counter
    latitude = Location();  // Set blank location in memory
    longitude = Location(); // Set blank location in memory
}

void Yacht::getPos()
{
    std::cout << "Enter the location of ship #" << serialNumber << ":\n"; // Prompt
    latitude.getPos();  // Gets location values for degrees minutes direction for latitude
    longitude.getPos(); // Gets location values for degrees minutes direction for longitude
}

void Yacht::display()
{
    // Displays the formatted yachts position
    std::cout << "The ship serial number is: " << serialNumber << std::endl;
    std::cout << "The position of ship #" << serialNumber << " is ";
    latitude.display(); // Latitude in correct format
    std::cout << " Latitude\t";
    longitude.display(); // Longitude in correct format
    std::cout << " Longitude\n";
}
```

Now it was time to put everything together in the main function. I decided I was going to use for loops to create and execute the code each stage of the program. I also realised that we didn't want a magic variable so I added a yacht count into the start of the program along with a vector

```
int main()
{
    Yacht::resetCounter(1); // Resets the static int counter
    int yachtsAmount = 3;
    std::vector<Yacht*> yachts;
```

containing pointers to Yachts to allow easy management for the Yacht throughout the programs lifetime.

Next was the main two core loops for creating and getting a latitude and longitude, then displaying them in the race as defined in the brief.

```
// Create the yachts
for (int i = 0; i < yachtsAmount; i++) {
    std::cout << "****************************************\n";
    Yacht* yacht = new Yacht();
    yachts.push_back(yacht); // Add instance to list
    yacht->getPos(); // Gets user input for Lat/Long position
}

std::cout << "\n*******Welcome to Ocean Race 2021-22*******\n\n";

// Loop through list and displays the yachts position
for (Yacht* yacht : yachts) {
    yacht->display(); // Display the position
    std::cout << std::endl; // Spacing in console
}
```

This was made simple enough as the functions were all contained in the Yacht class and made simple work of performing these functions.

Then we had to delete all the yachts memory as we used the 'new' keyword:

```
// Delete the no longer required memory before exiting
for (Yacht* yacht : yachts) {
    delete(yacht);
    yacht = nullptr;
}
```

I decided to test the program to see the results but found that the program would just exit before you could see the results when you opened the .exe directly. Debug mode allowed you to see the output but we don't know that all persons opening the .exe will be using debug mode in Visual Studio 2019. So I decided to add on last bit of code to the end of the main function.

```
// Stop the console from exiting straight away
std::cout << "Press any key to continue...";
char ingored = std::getchar(); // character is ignored / not used
return 0;
```

Completed main function

```
int main()
{
    Yacht::resetCounter(1); // Resets the static int counter
    int yachtsAmount = 3;
    std::vector<Yacht*> yachts;

    std::cout << "***********Ocean Race 2021-22***********\n\n";

    // Create the yachts
    for (int i = 0; i < yachtsAmount; i++) {
        std::cout << "****************************************\n";
        Yacht* yacht = new Yacht();
        yachts.push_back(yacht); // Add instance to list
        yacht->getPos(); // Gets user input for Lat/Long position
    }

    std::cout << "\n*******Welcome to Ocean Race 2021-22*******\n\n";

    // Loop through list and displays the yachts position
    for (Yacht* yacht : yachts) {
        yacht->display(); // Display the position
        std::cout << std::endl; // Spacing in console
    }

    // Delete the no longer required memory before exiting
    for (Yacht* yacht : yachts) {
        delete(yacht);
        yacht = nullptr;
    }

    // Stop the console from exiting straight away
    std::cout << "Press any key to continue...";
    char ingored = std::getchar(); // character is ignored / not used
    return 0;
}
```

With the program put together I debugged it and ensured the program handled all types of inputs into the console correctly. The code output is shown in the next section of the document.

Code Screen Shots

Task 1

```
***********Ocean Race 2021-22************

****************************************
Enter the location of ship #1:
Input the degrees between 0 and 180: 120
Input the minutes between 0 and 60: 45
Input direction (E/W/N/S): E
Input the degrees between 0 and 180: 34
Input the minutes between 0 and 60: 56
Input direction (E/W/N/S): N
****************************************
Enter the location of ship #2:
Input the degrees between 0 and 180: 34
Input the minutes between 0 and 60: 12
Input direction (E/W/N/S): w
Input the degrees between 0 and 180: 78
Input the minutes between 0 and 60: 34
Input direction (E/W/N/S): S
****************************************
Enter the location of ship #3:
Input the degrees between 0 and 180: 179
Input the minutes between 0 and 60: 23
Input direction (E/W/N/S): E
Input the degrees between 0 and 180: 126
Input the minutes between 0 and 60: 45
Input direction (E/W/N/S): S

*******Welcome to Ocean Race 2021-22*******

The ship serial number is: 1
The position of ship #1 is 120°45'E Latitude    34°56'N Longitude

The ship serial number is: 2
The position of ship #2 is 34°12'W Latitude     78°34'S Longitude

The ship serial number is: 3
The position of ship #3 is 179°23'E Latitude    126°45'S Longitude

Press any key to continue...
```

```
***********Ocean Race 2021-22************

****************************************
Enter the location of ship #1:
Input the degrees between 0 and 180: 181
Invalid input...
Input the degrees between 0 and 180: e
Invalid input...
Input the degrees between 0 and 180: 1
Input the minutes between 0 and 60:
```

Task 2

```
        CHARACTER CREATION
Which of the following would you like?
        1. Create a Warrior!
        2. Create a Priest!
        3. Create a Mage!
        4. Finish creating player characters!
1

Which race do you want?
        1. Human!
        2. Elf!
        3. Dwarf!
        4. Orc!
        5. Troll!
1
What would you like to name your character? : Alex

        CHARACTER CREATION
Which of the following would you like?
        1. Create a Warrior!
        2. Create a Priest!
        3. Create a Mage!
        4. Finish creating player characters!
2

Which race do you want?
        1. Human!
        2. Elf!
        3. Dwarf!
        4. Orc!
        5. Troll!
2
What would you like to name your character? : Mark

        CHARACTER CREATION
Which of the following would you like?
        1. Create a Warrior!
        2. Create a Priest!
        3. Create a Mage!
        4. Finish creating player characters!
3

Which race do you want?
        1. Human!
        2. Elf!
        3. Dwarf!
        4. Orc!
        5. Troll!
3
What would you like to name your character? : Raghii

        CHARACTER CREATION
```

```
       CHARACTER CREATION
Which of the following would you like?
       1. Create a Warrior!
       2. Create a Priest!
       3. Create a Mage!
       4. Finish creating player characters!
4
---------------
WARRIORS LIST:
---------------
I am a warrior with name Alex with race HUMAN and my attack is : I will destroy you with my sword, foul demon!

---------------
PRIESTS LIST:
---------------
I am a priest with name Mark with race ELF and my attack is : I will assault you with Holy Wrath!

---------------
MAGES LIST:
---------------
I am a mage with name Raghii with race DWARF and my attack is : I will crush you with the power of my arcane missiles!


Character Creation Done!.....
```

```
Which race do you want?
        1. Human!
        2. Elf!
        3. Dwarf!
        4. Orc!
        5. Troll!
0
Invalid input...

Which race do you want?
        1. Human!
        2. Elf!
        3. Dwarf!
        4. Orc!
        5. Troll!
e
Invalid input...
```

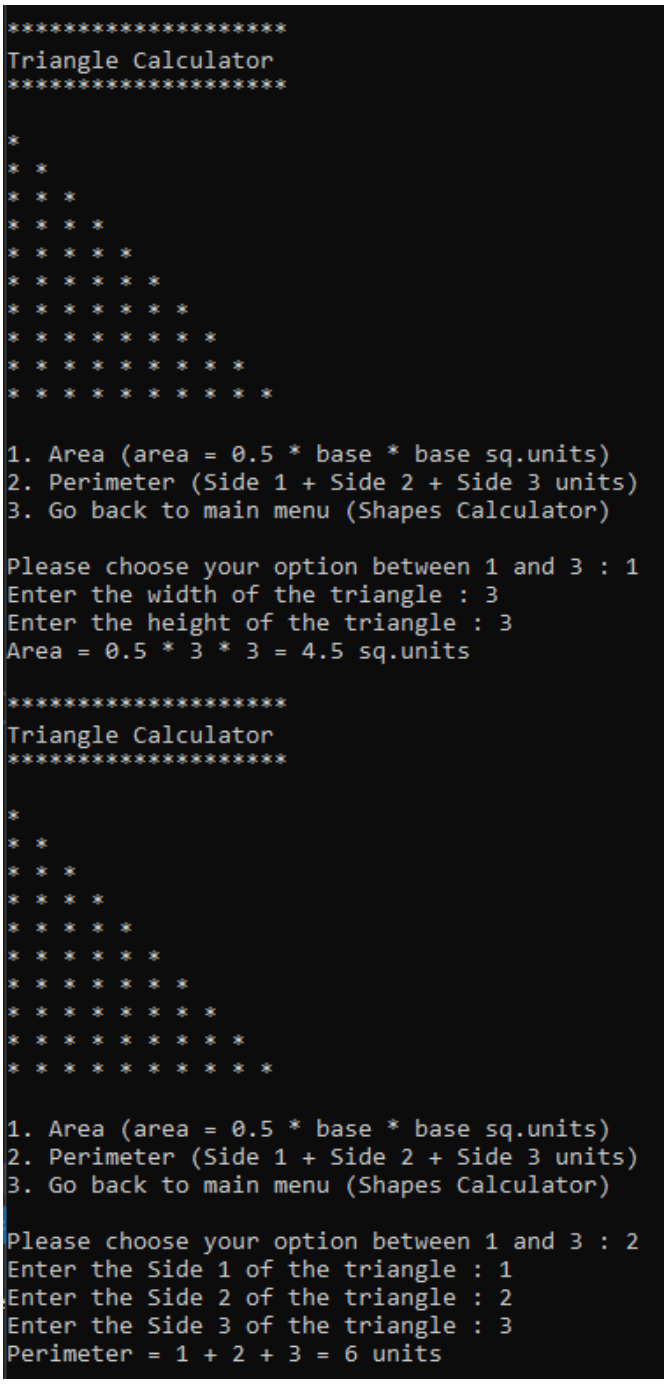
```
       CHARACTER CREATION
Which of the following would you like?
       1. Create a Warrior!
       2. Create a Priest!
       3. Create a Mage!
       4. Finish creating player characters!
;lasdj h780AT9237(GAOUS
Invalid input...
```

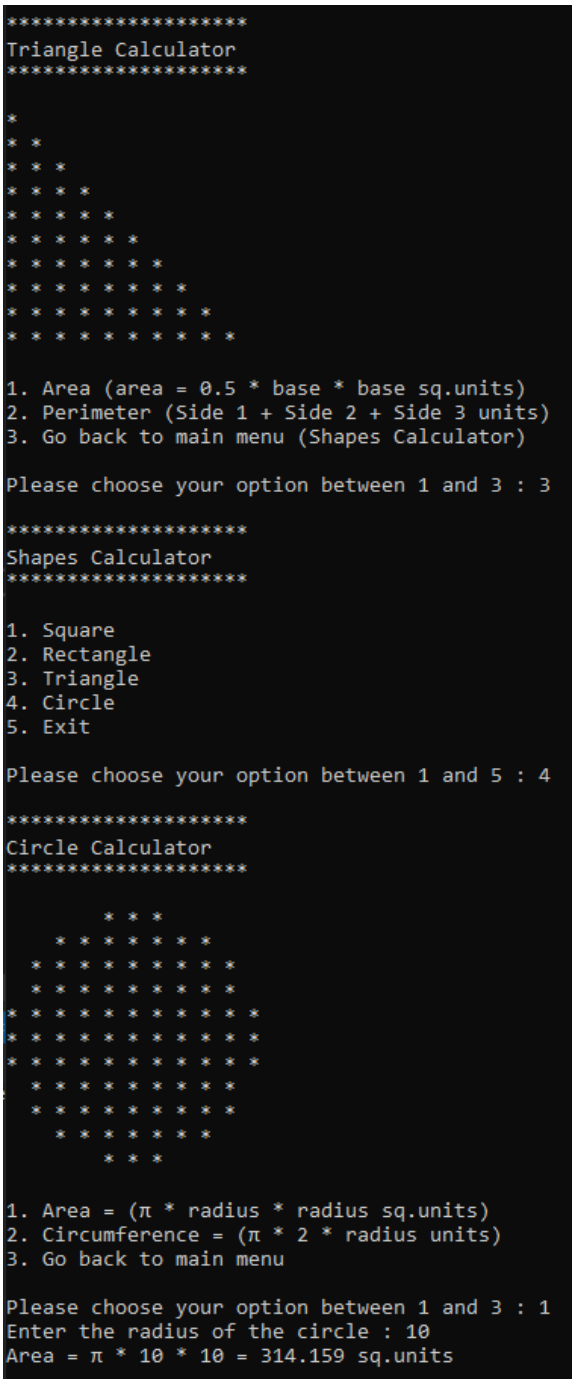
Task 3

```
**********
Shapes Calculator
**********

1. Square
2. Rectangle
3. Triangle
4. Circle
5. Exit

Please choose your option between 1 and 5 : 1

**********
Square Calculator
**********

* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *

1. Area (area = base * base sq.units)
2. Perimeter (perimeter = 4 * base units)
3. Go back to main menu

Please choose your option between 1 and 3 : 1
Enter the base of the square : 2
Area = 2 * 2 = 4 sq.units

**********
Square Calculator
**********

* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *

1. Area (area = base * base sq.units)
2. Perimeter (perimeter = 4 * base units)
3. Go back to main menu

Please choose your option between 1 and 3 : 2
Enter the base of the square : 2
Perimeter = 4 * 2 = 8 units
```

```
**********
Square Calculator
**********

* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *

1. Area (area = base * base sq.units)
2. Perimeter (perimeter = 4 * base units)
3. Go back to main menu

Please choose your option between 1 and 3 : 3

**********
Shapes Calculator
**********

1. Square
2. Rectangle
3. Triangle
4. Circle
5. Exit

Please choose your option between 1 and 5 : 2

**********
Rectangle Calculator
**********

* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *

1. Area (area = base * height sq.units)
2. Perimeter (perimeter = 2 * base + 2 * height units)
3. Go back to main menu

Please choose your option between 1 and 3 : 1
Enter the width of the rectangle : 1
Enter the height of the rectangle : 2
Area = 1 * 2 = 2 sq.units
```

```
**********************
Rectangle Calculator
**********************

* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *

1. Area (area = base * height sq.units)
2. Perimeter (perimeter = 2 * base + 2 * height units)
3. Go back to main menu

Please choose your option between 1 and 3 : 2
Enter the width of the rectangle : 1
Enter the height of the rectangle : 2
Perimeter = (2 * 1) + (2 * 2) = 6 units

**********************
Rectangle Calculator
**********************

* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *

1. Area (area = base * height sq.units)
2. Perimeter (perimeter = 2 * base + 2 * height units)
3. Go back to main menu

Please choose your option between 1 and 3 : 3

**********************
Shapes Calculator
**********************

1. Square
2. Rectangle
3. Triangle
4. Circle
5. Exit

Please choose your option between 1 and 5 : 3

**********************
Triangle Calculator
**********************
```


**********
Triangle Calculator
**********
1. Area (area = 0.5 * base * base sq.units)
2. Perimeter (Side 1 + Side 2 + Side 3 units)
3. Go back to main menu (Shapes Calculator)
Please choose your option between 1 and 3 : 1
Enter the width of the triangle : 3
Enter the height of the triangle : 3
Area = 0.5 * 3 * 3 = 4.5 sq.units
**********
Triangle Calculator
**********
1. Area (area = 0.5 * base * base sq.units)
2. Perimeter (Side 1 + Side 2 + Side 3 units)
3. Go back to main menu (Shapes Calculator)
Please choose your option between 1 and 3 : 2
Enter the Side 1 of the triangle : 1
Enter the Side 2 of the triangle : 2
Enter the Side 3 of the triangle : 3
Perimeter = 1 + 2 + 3 = 6 units


**********
Triangle Calculator
**********
1. Area (area = 0.5 * base * base sq.units)
2. Perimeter (Side 1 + Side 2 + Side 3 units)
3. Go back to main menu (Shapes Calculator)
Please choose your option between 1 and 3 : 3
**********
Shapes Calculator
**********
1. Square
2. Rectangle
3. Triangle
4. Circle
5. Exit
Please choose your option between 1 and 5 : 4
**********
Circle Calculator
**********
1. Area = (π * radius * radius sq.units)
2. Circumference = (π * 2 * radius units)
3. Go back to main menu
Please choose your option between 1 and 3 : 1
Enter the radius of the circle : 10
Area = π * 10 * 10 = 314.159 sq.units

```
**********************
Circle Calculator
**********************

        * * *
    * * * * * * *
  * * * * * * * * *
  * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
  * * * * * * * * *
  * * * * * * * * *
    * * * * * * *
        * * *

1. Area = (π * radius * radius sq.units)
2. Circumference = (π * 2 * radius units)
3. Go back to main menu

Please choose your option between 1 and 3 : 2
Enter the radius of the circle : 10
Circumference = π * 10 * 2 = 62.8319 units

**********************
Circle Calculator
**********************

        * * *
    * * * * * * *
  * * * * * * * * *
  * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
  * * * * * * * * *
  * * * * * * * * *
    * * * * * * *
        * * *

1. Area = (π * radius * radius sq.units)
2. Circumference = (π * 2 * radius units)
3. Go back to main menu

Please choose your option between 1 and 3 : 3

**********************
Shapes Calculator
**********************

1. Square
2. Rectangle
3. Triangle
4. Circle
5. Exit

Please choose your option between 1 and 5 : 5
```

```
********************
Shapes Calculator
********************

1. Square
2. Rectangle
3. Triangle
4. Circle
5. Exit

Please choose your option between 1 and 5 : 0
Invalid input...

Please choose your option between 1 and 5 : 1

********************
Square Calculator
********************

* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *

1. Area (area = base * base sq.units)
2. Perimeter (perimeter = 4 * base units)
3. Go back to main menu

Please choose your option between 1 and 3 : 0
Invalid input...

Please choose your option between 1 and 3 : e
Invalid input...

Please choose your option between 1 and 3 : 1
Enter the base of the square : e
Invalid input...

Enter the base of the square : -1
Invalid input...

Enter the base of the square :
```

Task 4

```
Main Menu
1. Create Alien Pairs
2. Create Alien Offsprings
3. Compare Alien Offspring Prestige
4. Exit
Please enter your option : 1

Aliens 1, 2, 3 and 4 created...

Main Menu
1. Create Alien Pairs
2. Create Alien Offsprings
3. Compare Alien Offspring Prestige
4. Exit
Please enter your option : 2

Aliens 5 and 6 created...

Main Menu
1. Create Alien Pairs
2. Create Alien Offsprings
3. Compare Alien Offspring Prestige
4. Exit
Please enter your option : 3

Offspring prestige comparison :
Alien 5 == Alien 6 ? false
Alien 5 != Alien 6 ? true
Alien 5 >  Alien 6 ? false
Alien 5 >= Alien 6 ? false
Alien 5 <  Alien 6 ? true
Alien 5 <= Alien 6 ? true

Main Menu
1. Create Alien Pairs
2. Create Alien Offsprings
3. Compare Alien Offspring Prestige
4. Exit
Please enter your option : 4
```

```
Main Menu
1. Create Alien Pairs
2. Create Alien Offsprings
3. Compare Alien Offspring Prestige
4. Exit
Please enter your option : 1

Aliens 1, 2, 3 and 4 created...

Main Menu
1. Create Alien Pairs
2. Create Alien Offsprings
3. Compare Alien Offspring Prestige
4. Exit
Please enter your option : 1
Invalid order - unable to perform operation...

Main Menu
1. Create Alien Pairs
2. Create Alien Offsprings
3. Compare Alien Offspring Prestige
4. Exit
Please enter your option : 2

Aliens 5 and 6 created...
```