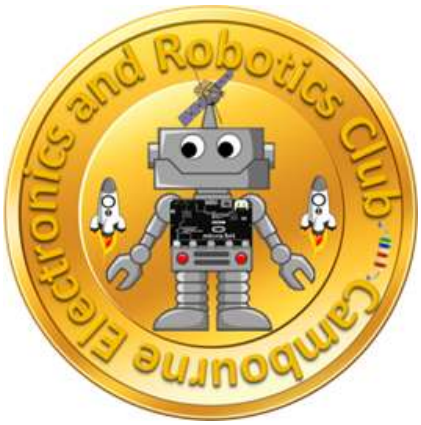


Ignite

CERC Newsletter

APRIL 2021

Shrobona
Founder

Sujit Bhattacharya
Mentor

Maya & Sreenidhi
Editors

Message from the Founder of CERC

Dr. Shrobona Bhattacharya

Not every town has their own Science Festival, Cambourne is lucky as it has its own Science Festival since 2016.

Over the years the number of participants increased, in 2020 Covid time 160 young scientists presented their scientific models and prototype online when we decided to go on Zoom.

It went for two days for 12 hours and children presented their works from 7 different time zones! We also renamed the festival from Cambourne Science festival to Cambridge Global Children Science Festival.

We have a plan to run both these festivals this year i.e., one to organise a science festival where children can present their prototypes and models at Cambourne and another one for the children from all over the world.

We not only encourage children for their demonstration at their personal level but also encourage them to act as an young science ambassador where they can motivate their classmates and friends to also take part in these festivals.

Message from the Mentor Dr. Sujit Bhattacharya

We are now coming to end of the Robotics and Python batch that we started in Jan 2021. For the Maths with Python, we had 24 students. For the Robotics batch we had registration of 52 students for Beginners, 30 students for Intermediate and 10 students for the advanced batch. All these students are graduating this week and moving to the next level.

We have seen tremendous potential in these kids and sure many of them will pursue a career in STEM. The computational thinking is very critical for the 21st century workforce in whatever profession they may choose. We are very impressed with the commitment of the kids and parents.

The kids have now completed more than 30-projects along with us in the Robotics Beginners course and they are now empowered to do several projects themselves. A cohort of **52 students** in age group of 7-15 years from Australia, US, India and UK had 10 weeks of mentoring in Science, Electronics and Robotics completing more than 40 projects and presenting more than 60 Young Scientist demonstrations during the course.

The graduation cum certification ceremony is planned on Sunday, 11th April from 3.00pm (London).

Please feel free to join the Zoom session:

Topic: Young Scientist Presentation & Certification

Time: Sunday, Apr 11, 2021 03:00 PM London

https://tinyurl.com/Young-Scientist-Presentation

Meeting ID: 870 7103 6674
Passcode: 513405

Please pass on the above link to all your friends, their classmates, teachers and family members to join for the presentation to appreciate the projects by our young budding scientists. This can go global.

We have planned several workshops on IoT, Artificial Intelligence and building a rover.

For more information on all the initiatives, please complete the form

https://tinyurl.com/cifstem-info

Message from Editorial board by Venkat Kommi

Hope everyone is enjoying Easter schools holidays and good weather from Spring.

It is nice to see different topics for this edition of Ignite. We are looking forward to seeing more response from the readers.

Readers please feel free to comment on their articles. This will boost their confidence levels.

The Ignite magazine is a community based volunteered magazine, where a group of professionals, academics and young children are working together.

These articles are written by the children and are edited by a team.

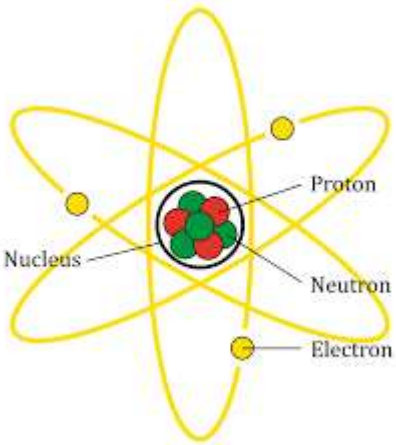
The editorial team is in charge of the publication of the magazine.

If you have any questions, suggestions, or concerns, please address them to ignite.camcare@gmail.com

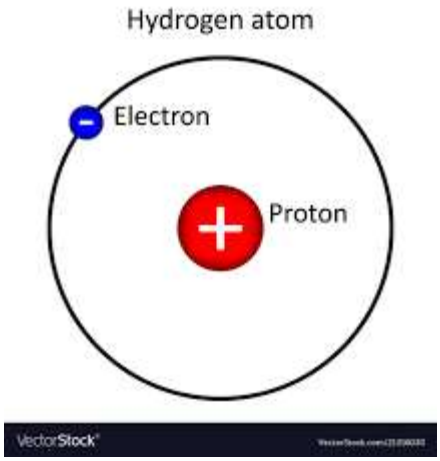
Why do Atoms form chemical bonds?

by Vivek Kommi, 12 years, Perse School, Cambridge,UK

Atoms form chemical bonds to make their outer electron shells more stable. The type of chemical bond maximizes the stability of the atoms that form it, for example, when a solid is formed the atoms are closer and therefore makes the outer shell more stable whereas in a liquid, the atoms are more spaced apart making the atoms have less structure. An ionic bond, where one atom essentially donates an electron to another, forms when one atom becomes stable by losing its outer electrons and the other atoms become stable (usually by filling its valence shell) by gaining the electrons for example sodium and chloride donates an electron around itself. Covalent bonds form when sharing atoms results in the highest stability for example water and diamonds. Other types of bonds besides ionic and covalent chemical bonds exist, too.



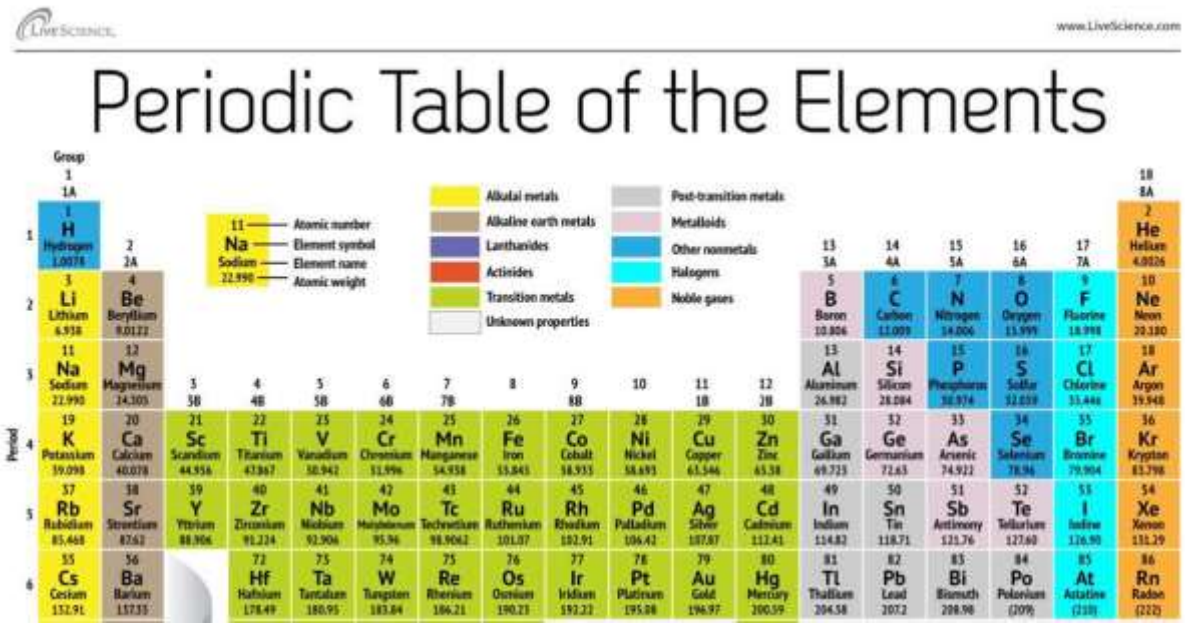
The very first electron shell only holds two electrons. A hydrogen atom (atomic number 1) has one proton and a lone electron, so it can readily share its electron with the outer shell of another atom. A helium atom (atomic number 2), has two protons and two electrons. The two electrons complete its outer electron shell (the only electron shell it has), plus the atom is electrically neutral this way. This makes helium stable and unlikely to form a chemical bond.



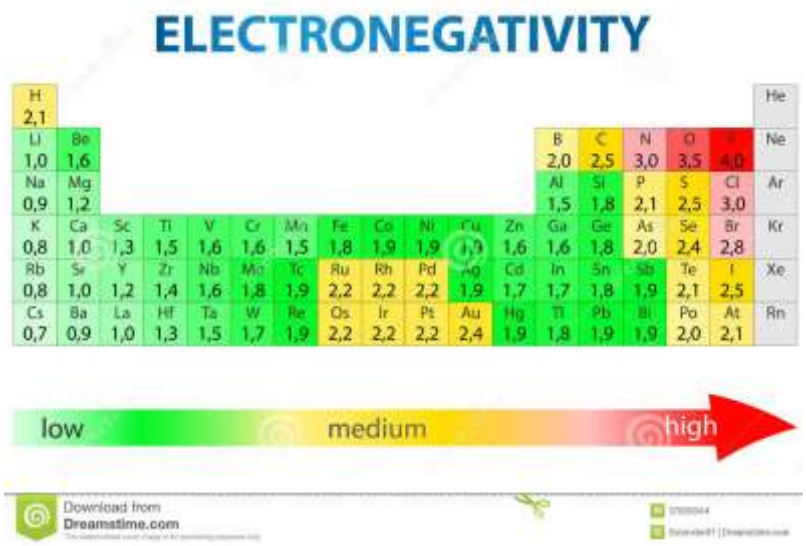
Most atoms need eight electrons to complete their outer shell. So, an atom that has two outer electrons will often form a chemical bond with an atom that lacks two electrons.

For example, a sodium atom has one lone electron in its outer shell. A chlorine atom, in contrast, is short one electron to fill its outer shell. Sodium readily donates its outer electron (forming the Na+ ion, since it then has one more proton than it has electrons), while chlorine readily accepts a donated electron (making the Cl- ion, since chlorine is stable when it has one more electron than it has protons). Sodium and chlorine form an ionic bond with each other to form table salt (sodium chloride).

You can use the periodic table to make several predictions about whether atoms will form bonds and what type of bonds they might form with each other. On the far right-hand side of the periodic table is the group of elements called the noble gases Atoms of these elements (e.g., helium, krypton, and neon) have full outer electron shells. These atoms are stable and very rarely form bonds with other atoms.



One of the best ways to predict whether atoms will bond with each other and what type of bonds they will form is to compare the electronegativity values of the atoms. Electronegativity is a measure of the attraction an atom has to electrons in a chemical bond.



A large difference between electronegativity values between atoms indicates one atom is attracted to electrons, while the other can accept electrons. These atoms usually form ionic bonds with each other. This type of bond forms between a metal atom and a nonmetal atom. If the electronegativity values between two atoms are comparable, they may still form chemical bonds to increase the stability of their valence electron shell. These atoms usually form covalent bonds. Electronegativity increases as you move from left to right across the periodic table (except for the noble gases). It decreases as you move down a column or group of the table. Atoms on the left-hand side of the table readily form ionic bonds with atoms on the right side. Atoms in the middle of the table often form metallic or covalent bonds with each other.

Google Foo Bar Challenge *by Avni Balan, 13 years, IVC School, Histon,UK*

Welcome back to Google Foobar Challenges. This is the second out of a series, so if you're new here, I'll put a link to the first part of the series at the end. So far in this story, we've successfully automated a labour shift removal program to get Commander Lambda to approve of us, so that we can later gain her trust, gain access to the LAMBCHOP, destroy it and save Bunny Planet. However, so far our efforts have gone unnoticed by Lambda, so we've still got quite a bit of impressing left to do. We've been faced with the task of repairing the ship's solar panels, and writing a program to figure out which panels to take offline to repair while keeping the other panels in loop. Let's see how well we did with this.

foobar #2: Power Hungry – Solution

We were tasked with removing certain elements – for example, 0s and negative numbers – to get the maximum power output, or product, of an array. This is just fancy language for removing numbers from a list so that when we multiply what's left of it, we get the highest possible number that you can get through multiplying any numbers from that array. I gave you test cases that you should've tested your code against to see if it worked. Here's a picture of my code, since typing it out didn't work last time:

```
def solution(xs):

    if len(xs) == 1:
        return str(xs[0])

    if any(i<0 for i in xs):
        negs = [i for i in xs if i < 0]
        if len(negs) % 2 != 0:
            xs.remove(max(negs))

    if any(i==0 for i in xs):
        xs = [i for i in xs if i != 0]

    result = 1
    for i in xs:
        result *= i
    if len(xs) == 0:
        return '0'
    else:
        return str(result)
```

How this code works is first it checks if there is only one element in the list given (if len(xs) == 1:) and that if there is, just return that (return str(xs[0])). This happens because if it were given a list of just one negative number, the largest product to return from that wouldn't be 0, it would be just that number.

If that's all false, it checks if there are any negative numbers in the list given (if any(i<0 for i in xs):) If there are, it lists all the negative numbers and puts them in a list called negs (negs = [i for i in xs if i < 0]) Then it checks if the number of negatives is even (if len(negs) % 2 != 0:), since if it is even, there won't be any need to remove any, since when you multiply an even amount of negatives together, the answer will always be positive. But if it isn't, then the answer will be negative, which we don't want. So if the amount is odd, it removes the closest negative number to 0, or the largest number out of the group, from the list xs (xs.remove(max(negs))), since that number will make the least difference to the product. Then the code checks if there are any 0s in xs (if any(i==0 for i in xs):) and removes them (xs = [i for i in xs if i != 0]).

Finally, after removal, it multiplies all the remaining elements with the number 1 and stores it in a variable called 'result' (result = 1 / for i in xs: // result *= i). Then it checks if actually, there's nothing in xs, which I realise we could've done from the start, so we just return 0 (if len(xs) == 0: // return '0') and if that's wrong, return the variable 'result' (else: // return str(result)).

So that was my code – how did yours work? Here's just a couple more cases to test it against, called edge cases. These are worst-case scenarios, which while are rare, can be very hard to deal with when they come along:

[-3]

Output: '-3'

[-4, -8]

Output: '32'

[0, -1, 0]

Output: '0'

Now with that all out of the way, here's the next problem for you to have a go with if you can:

foobar #3: Gearing Up to Destruction

As Commander Lambda's personal assistant, you've been assigned the task of fixing some of the LAMBCHOP doomsday device's gears in place. It should be simple - just add gears to the pegs – some smaller, some bigger, depending on what you want the output to be. But the problem is, due to the layout of the LAMBCHOP and the complicated system of beams and pipes supporting it, the pegs that will support the gears are fixed in place. The LAMBCHOP's engineers have given you lists of numbers identifying which position pegs have been placed along various support beams. You need to place a gear on each peg (otherwise the gears will collide with unoccupied pegs).

The engineers have plenty of gears in all different sizes stocked up, so you can choose gears of any size, from a radius of 1 on up.

Your goal is to build a system where the last gear rotates at twice the rate (in revolutions per minute, or rpm) of the first gear, no matter the direction. This means the radius of the last gear has to be half the radius of the first gear. Each gear (except the last) must touch the gear on the next peg to the right to turn it.

Your job is to find the radius of the first gear and to represent it as a fraction - in the format [numerator, denominator] - in its simplest form. For example, if the radius is 1.5, represent it as [3, 2] from the fraction 3/2, or if you get 2/4, instead return [1, 2] from the fraction 1/2. However, for some orders of pegs it may be impossible to find correctly sized gears, s when it comes to that, return [-1, -1]. For example, for the list [4, 30, 50] your function should return the list [12, 1], since the second gear will have a radius of 14 and the last a radius of 6, which is half of 12. Try your code against these cases:

[4, 30, 50]

Output: [12, 1]

[4, 17, 50]

Output: [-1, -1]

[4, 8, 9, 11]

Output: [10, 3]

You can look things up if you need help, but don't look up the solution code without learning anything.

My Github and Gmail accounts:

avni.k.balan@gmail.com

https://github.com/not-that-python

Link to Feb IGNITE Issue / first part of the series:

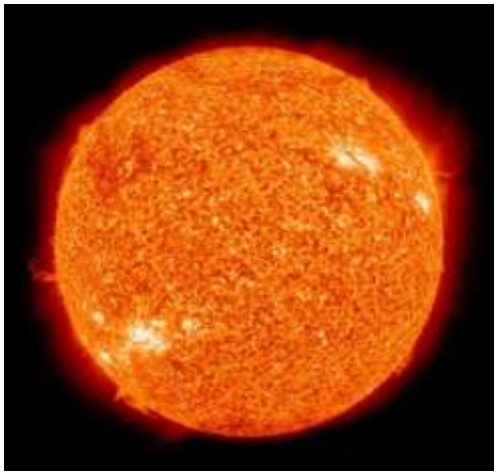
https://www.flipsnack.com/camcare/cerc-ignite-feb.html

My name is Avni. I'm 12 years old and I really like to code. I'm mainly Python-based, but I also use p5.js. It's really fun to look into computing logic and the kinds of problems you can solve. I'm also interested in law and a little bit of politics. Someday, I'd like to somehow participate in court and look a little further into magistrate or high court, as well as brush up my debating skills.

The Sun

by Sairam Batchu, 13 years, Perse School, Cambridge, UK

The Sun is the largest mass, within the solar system, and accounts for 98.86 percent of all mass comprising the Solar System. The Sun is what is known as main sequence star, a mass that mainly contains hydrogen and helium. It is a sphere, which fuses hydrogen in such a way that it releases a vast amount of energy, which allows the star from collapsing in on itself. The way that the Sun is in perfect harmony, meaning that it is a stable star, is because the Sun can fuse elements at a rate that it is able to counter the forces of gravity, while also fusing elements that allow the Sun to radiate heat and light. Since the Sun is described as a main-sequence star, there are certain conditions that the Sun must meet to be under the category of a main sequence star, the first of which being that the Sun's mass must fall within 1.4 * 10^29 kg and 3.0*10^32 kg. This range dictates, in certain conditions, how a star can behave. This is usually also known as the range of mass between 75 times the mass of Jupiter and no more than 150 times the wight of the Sun itself. By knowing the information of the category of which the sun falls in (according to mass) scientists can predict the life cycle of a star, like the fact that the Sun, at the end of its life, will not explode into a supernova, but rapidly expand into another phase of its life being a red giant.



To be classified as a main-sequence star, the mass must have nuclear fusion occurring, which is the fusion of elements to keep the mass stable. The way that a star reaches the end of its life is when the star fuses elements that do not give enough energy to keep the balance between gravity and expansion. After a certain period of the Sun's life, the Sun fuses elements heavier than hydrogen and helium, and since they are heavier, they do not output as much energy when being fused. After the Sun runs out of fuel to sustain the balance, which is about 4 billion years from now, the Sun will expand into a red giant, since the core of the star is contracting because of gravity, but the outer layers like the corona are still expanding due to hydrogen fusion within those layers. The sun gets larger, redder, and extremely bright as it cools. When the sun expands into the red-giant phase, the goldilocks zone that had allowed the Earth to sustain life, the balance of heat, will be destroyed as it expands to engulf mercury and place Venus in and orbital trajectory that ensures that it will crash into the Sun.

The fate of the Earth is that the intense heat will kill all life and solar flares and extreme solar winds shall ruin the magnetic field, that protects us from solar winds, and the lack of a magnetic field means that the atmosphere shall vanish.

At first, Earth will transform into the condition of Venus and after the destruction of the atmosphere, it will evolve into an extremely volcanically active version of Mars, without the inclusion of any liquid water or ice. After the sun evolves into a red giant, the outer layers of the Sun will float away, since the forces acting on the Sun's mass are not strong. When the Sun is in the Red giant phase, Jupiter and Saturn will fall into the goldilocks' zone and life may be sustained on its moons. Then, it will transition into a red supergiant and the chances of life existing in this solar system at this rate will be extremely low. After that, all the outer layers of the sun will float away as mass that will then form another planetary nebulae and the sun's core will remain as a white dwarf, which can barely exert enough gravitational pull to hold in the rocky planets.

The Sun's diameter is roughly 870,000 miles, meaning about 110 Earths could line up and touch the edges of the Sun. In terms of sheer volume, 1.4 million Earths could fit inside the Sun, when effectively organised. Since the Sun will transform into a red giant in about 130 million years, the sun will stop burning hydrogen and engulf the Earth, just before it transitions into becoming mass for planetary nebulae. The Sun is almost a perfect sphere, with only a 10 km difference between its polar and equatorial diameters, the closest object to a sphere observed in nature. The white dwarf that the sun's core will eventually be left as will be roughly the size of the Earth now. The Sun's core temperature reaches about 15 million degrees. The reason that the Sun generates solar winds is because at the corona, or the outermost layer of the Sun, sheds away hydrogen in burst, which hurtle through the solar system at a speed which hits and breaks through the magnetic fields of the Earth and generates the Aurora Borealis, or the Northern Lights. The sun if often referred to as a GV star, or a star that is a yellow dwarf. It is close to impossible to live on the sun since it reaches about 4000 degrees Celsius on the surface. It **would take** 1,430,769 hours to drive there at 65 miles per hour. It **would take** 59,615 days to drive there at 65 miles per hour.

Google Foo Bar Challenge *by Aaron Balan, 11 years, Histon,UK*

In the February magazine, I wrote an article about my first problem in the foobar challenge. This month, I will write about how I completed my second challenge. I will then end this article with my third challenge that you can ponder on and maybe solve. I will reveal my solution to this one in the next issue and give you another problem to work on and we will continue like this in the following issues. Hope you enjoy my articles and solving the problems in the coming months.

Do any of you remember the problem I gave you a while ago? Well done if you do! And fantastic if you solved it! For anyone who doesn't remember it, I recommend going to the link above. Here is the way I solved it, but it is probably going to need a little explaining:

```
def fib(n):
    if n == 0 or n == 1:
        return 1
    return fib(n-1) + fib(n-2)
def solution(total_lambs):
    max_lst = []
    min_lst = []
    i = 0
    while sum(max_lst) >= total_lambs:
        max_lst.append(2**i)
        i += 1
    i = 0
    while sum(min_lst) >= total_lambs:
        min_lst.append(fib(i))
        i += 1
    return len(min_lst) - len(max_lst)
```

Here, the first function I write isn't actually the solution, but a different function. This function returns the n^{th} number in the Fibonacci sequence. Next, I define the actual function that produces the answer to the problem. Now, the way I solved this could be considered to be cheating by some people, because it doesn't follow the logic of the problem, and is only solved through a simple emergent behaviour: rule 2 and 3 of the 4 key rules of this problem makes the maximum amounts of the LAMBs that Commander Lambda can give are a list of the powers of 2, and the least she can give are a list of the Fibonacci sequence. So most of this code is just getting the Fibonacci numbers and the powers of 2. The last line returns what the question is looking for, which is the difference between the number of henchmen she can pay, while paying the maximum amount and the minimum amount.

I had completed the first challenge of this level, but it wasn't over yet! I still had one more challenge until I could move to level 3. Hope you have fun maybe attempting this problem, and here it is:

In order to destroy Commander Lambda's LAMBCHOP doomsday device, you'll need access to it. But the only door leading to the LAMBCHOP chamber is secured with a unique lock system whose number of passcodes changes daily. Commander Lambda gets a report every day that includes the locks' access codes, but only she knows how to figure out which of several lists contains the access codes. You need to find a way to determine which list contains the access codes once you're ready to go in.

Fortunately, now that you're Commander Lambda's personal assistant, she's confided to you that she made all the access codes "lucky triples" in order to help her better find them in the lists. A "lucky triple" is a tuple (x, y, z) where x divides y and y divides z, such as (1, 2, 4). With that information, you can figure out which list contains the number of access codes that matches the number of locks on the door when you're ready to go in (for example, if there's 5 passcodes, you'd need to find a list with 5 "lucky triple" access codes).

Write a function solution(l) that takes a list of positive integers l and counts the number of "lucky triples" of (li, lj, lk) where the list indices meet the requirement i < j < k. The length of l is between 2 and 2000 inclusive. The elements of l are between 1 and 999999 inclusive. The answer fits within a signed 32-bit integer. Some of the lists are purposely generated without any access codes to throw off spies, so if no triples are found, return 0.

For example, [1, 2, 3, 4, 5, 6] has the triples: [1, 2, 4], [1, 2, 6], [1, 3, 6], making the answer 3 total.

Have a good think about it and I will see you soon! :)

My name is Aaron and I'm 11 years old. I really like coding in Python and JavaScript (mainly p5.js for JavaScript) and I am really passionate about maths, my favorite subjects being calculus with derivatives and integrals, and linear algebra with vectors and matrices. I must thank Sujit and the CERC team for showing me how to combine complex maths and complex coding together to create wonderfully cool projects.

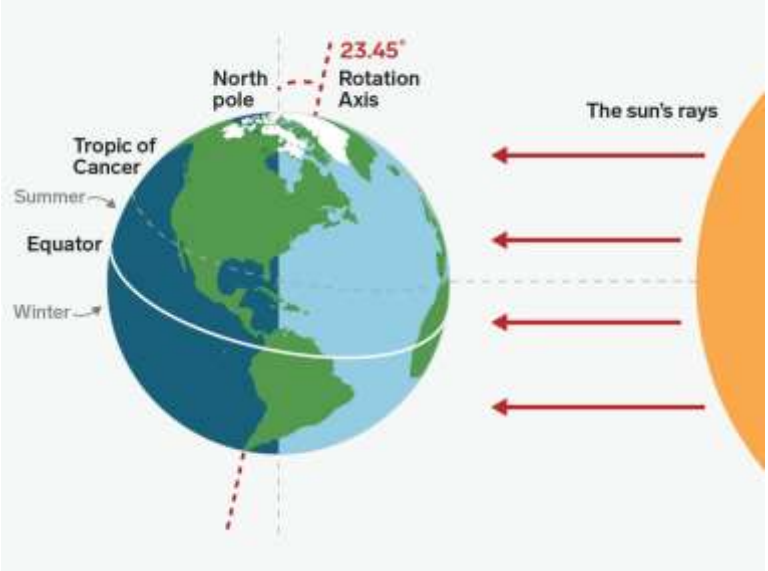
Stars and Nebula *by Leo Fell, 9 years, Cambourne, UK*

As well as learning about rockets and space travel, I have been learning about how the stars appear to rotate around Polaris (The North Star), because Earth is spinning. Earth rotates all the way around in one day and at 15 degrees per hour. The North Star is pointing to the North Pole instead of straight up.

The first picture shows that because the earth rotates, the stars move as we look at them. This picture is two hours of the stars rotating in the middle of the night whilst I was fast asleep. The North Star is the one in the middle. It is still a dot because it is lined up with the Earth's rotation axis.

The second picture taken looking to the South-West. You can see the stars going East (on the left) to West (on the right) and you can see the star trails are tilted at an angle because Earth is on its tilted axis as it rotates around the sun.

Earth's tilt is about 23 degrees as you can also see in the picture below.



In the night sky, the star patterns all have names. The Orion star constellation is circled in red. You can see Orion's belt which are the three stars in the middle. Below them is Orion's sword and you can just see Orions Nebula in the middle. The star at the top left is called Betelgeuse. Betelgeuse is a red supergiant star about 950 times the size of our sun and the 10th brightest star in the sky. It is burning fast and on its way to turning into a supernova when it burns out in a few million years.

In the green circle is Mars. This picture was taken just after NASAs Perseverance rover had landed on the surface.

This a picture of Orion's nebula (seen in Orion's sword in the previous picture) taken through a telescope. Because it is so far away this is what it looked like 1344 years ago, even though the picture was taken the other day. Even then, it is still the closest nebula to Earth in our galaxy.

A nebula is a made up of gasses and dust and it is where new stars are born.

Article from the Young Scientists for the Next Issue:

Ignite Team is looking for young scientists to write an article for the next issue of the magazine. Please submit the article by ***Sunday, 2nd May 2021*** *by emailing the article at ignite.camcare@gmail.com and with a subject "Article for Ignite".*

Please use the following template for writing the article:

https://bit.ly/3nF820h

Volunteers Needed:

Ignite Team is looking young volunteers with knowledge of Editing/Presentation Skills, HTML, Web Authoring tools.

Volunteering certification can also be provided for the Duke of Edinburgh Skill enhancement.