PROBLEM OF THE MONTH POM **October, 2021**

2021 - 2022

MATHEMATICS

5 points:
How many distinct (non-congruent) rectangles with integer sides have area equal to 2048?

Answer:
6

Solution:
The problem is that of splitting the total of 11 factors of 2 between the two sides. However, once we choose the factor 2^n applied to the first side, the factor applied to the second side is determined, too, it is 2^{11-n}. There are 12 possible ways to choose the factor applied to the first side (2^0, 2^1, …, 2^{11}). However, for each of these choices there is also one where the sides are swapped, corresponding to a rotation of the rectangle. Therefore, we need to divide the result by 2, which gives 12/2=6.

10 points:
How many distinct (non-congruent) rectangular cuboids with integer sides have volume equal to 2048?

Answer:
16

Solution:
A cuboid with the volume 2048 = 2^{11} is obtained from a cube 1x1x1 by multiplying the length of each side by a power of 2, such that the total number of factors of 2 applied to all three sides is 11 and the volume is multiplied by 2^{11}. Different cuboids are obtained by a different choice of the number of factors 2 (out of 11 total) that are applied to each side. This is the problem of putting n=11 identical objects (factors of 2) into m=3 identical boxes (corresponding to three sides). The problem is easily solved for the case where boxes are distinct and can be arranged from left to right. We thus obtain a set of n+m-1 positions filled either with an object (factor of 2), or a wall separating the adjacent boxes; there are 2 outer walls and m-1 internal walls that are common to the adjacent boxes. Different fillings of boxes correspond to different choices of the positions for the m-1 internal walls of the boxes (two outer walls are fixed). The answer to this problem is ${}_{n+m-1}C_{m-1}$ (“from n+m-1 choose m-1, order does not matter”). For m=3 boxes and n=11 factors of 2 (objects), the answer is ${}_{13}C_2$ = 13!/11!/2! = 78.

For the case of identical boxes, this result should be corrected because splits that correspond to permuting the boxes (i.e. where the lengths of the two different sides of the obtained cuboid are simply interchanged, corresponding to an identical rotated cuboid) should only be counted once. One way to correct for the overcount, is to first make sure that every split is counted 3!=6 times and divide the corrected result by 3! Splits where all 3 boxes have different filling were all counted 3! times. This, however, is not the case for a split where two boxes are equally filled (corresponding to a cuboid with one face that is a square), which were only counted 3 times; we need to add 3 for each such case. There are [n/2]+1=6 such cases (2x0, 2x1, 2x2, …, 2x5), so we need to add 18. In principle, if n would be divisible by 3, we would also need to add 5 to account for the fact that an equal split was only counted once, but 11 is not divisible by 3, so we add 3x6=18. Hence, the final answer is (78+18)/3! = 16.

PHYSICS

5 points:

You are standing on a skateboard (total mass M), and have a handful of marbles in your pocket. Total mass of the marbles is m, and $m << M$. You throw one marble at a time horizontally, in one direction, with speed v. How fast will you be going once you are out of marbles?

Answer:

mv/M.

Solution:

The initial velocity is 0. Because the marbles are much lighter than the skater, we can neglect the change of mass of the skater. Simple conservation of energy dictates that $mv = Mv_{skater}$.
Note that for the same reason that we can neglect the change of velocity of the skater, we can say that the speed of marbles in the rest frame of the ground is approximately the same as in the skater's frame (v_{skater} is much smaller than v).
As an aside, without the condition $m << M$, this would be a much harder problem, corresponding to the rocket propulsion dynamics:
https://en.wikipedia.org/wiki/Tsiolkovsky_rocket_equation

10 points:

It turns out that light carries momentum! If a light beam has energy E, then the magnitude of its momentum p is given by $E = cp$, where c is the speed of light.
Hypothetical methods of transport known as 'solar sails' take advantage of this, similar to how sails of sea ships take advantage of the momentum carried by wind.
Suppose a solar sail of area A and mass m is orbiting around the sun at radius R. What acceleration does it experience due to the light from the sun, if the sail is oriented to face the incoming light? Assume the sun emits total power P. Note that the solar sail is made of material that absorbs all light.
How will your acceleration change if the solar sail is made of a reflective material?

Answer:

$(PA/(4\pi R^2 c m)$

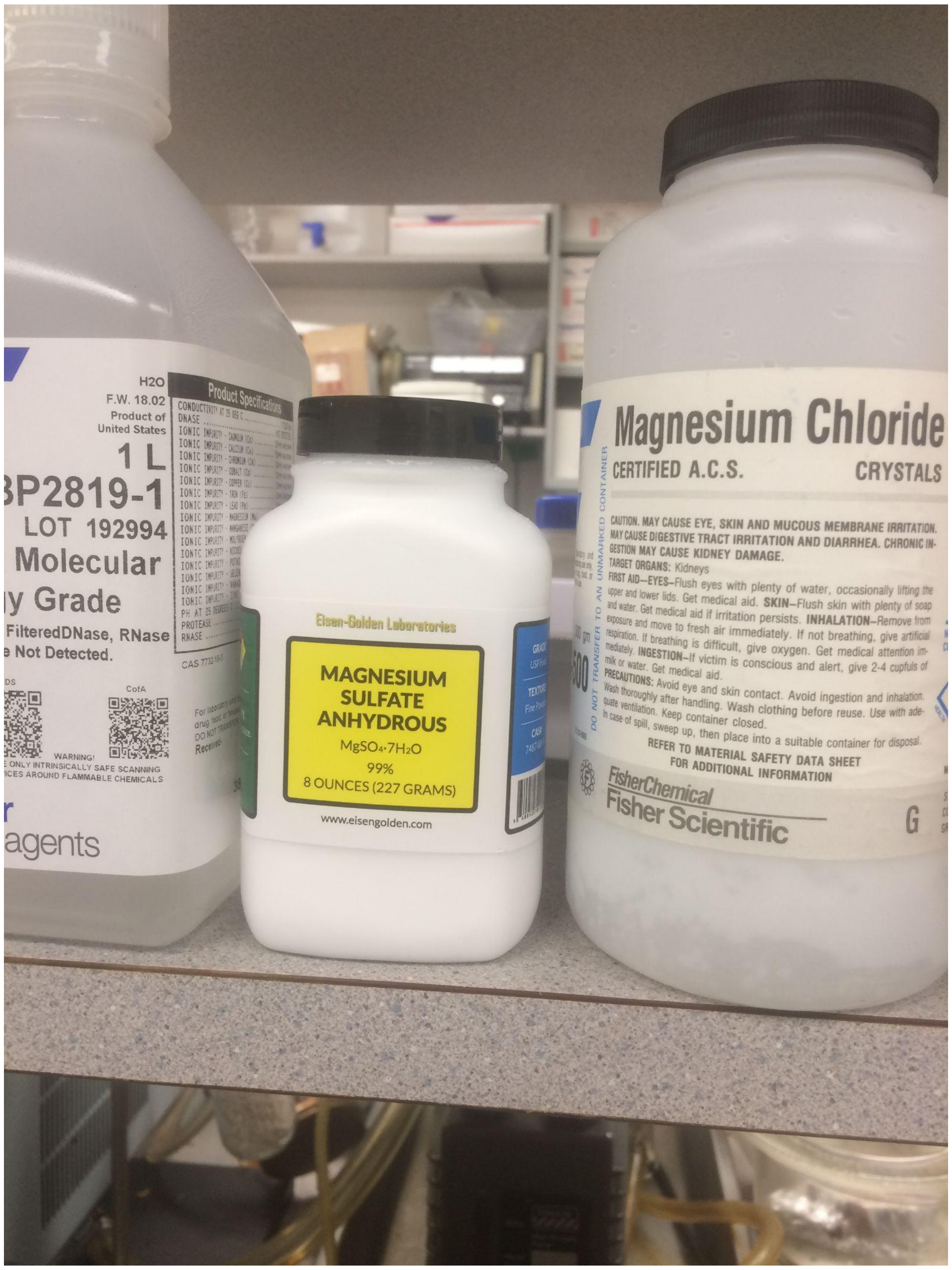
Solution:

The fraction of solar power that hits the sail is $A/(4\pi R^2)$. Power is energy per second. So $PA/(4\pi R^2 c)$ is the imparted momentum per second, which is the force on the sail. From F=ma, then the acceleration on the sail is $PA/(4\pi R^2 c m)$
If the sail is reflective, instead of stopping the light particles through absorption, it will send them back with the same speed in the opposite direction, so the amount of momentum transfer will double. So, the acceleration will also double.

CHEMISTRY

5 points:

The photo (see below) shows a shelf in Mark's lab. Each bottle contains chemicals purchased from certified suppliers. Mark took this photo because some of the labels cause what is commonly referred to as "cognitive dissonance". Explain what is wrong with the label, and propose the experiment that would allow Mark to resolve the problem.


H2O
F.W. 18.02
Product of United States
1 L
BP2819-1
LOT 192994
Molecular
Grade
FilteredDNase, RNase
Not Detected.
Product Specifications
agents
Eisen-Golden Laboratories
MAGNESIUM SULFATE ANHYDROUS
$MgSO_4 \cdot 7H_2O$
99%
8 OUNCES (227 GRAMS)
www.eisengolden.com
Magnesium Chloride
CERTIFIED A.C.S.
CRYSTALS
CAUTION. MAY CAUSE EYE, SKIN AND MUCOUS MEMBRANE IRRITATION. MAY CAUSE DIGESTIVE TRACT IRRITATION AND DIARRHEA. CHRONIC INGESTION MAY CAUSE KIDNEY DAMAGE.
TARGET ORGANS: Kidneys
REFER TO MATERIAL SAFETY DATA SHEET FOR ADDITIONAL INFORMATION
DO NOT TRANSFER TO AN UNMARKED CONTAINER
FisherChemical
Fisher Scientific

Answer:

The label that causes cognitive dissonance is on the middle bottle. It says “Magnesium sulfate anhydrous”, but the formula says the opposite. This formula belongs to a so-called “crystal hydrate”. This type of crystals are usually formed by some salts when they precipitate from an aqueous solution. In these crystals, water molecules form stable bonds (hydrogen bonds or donor-acceptor bonds) with salt’s anions and/or cations, thereby becoming an integral part of the crystal lattice. Sometimes, crystal hydrates form bigger and more beautiful crystals than anhydrous salts, and the properties of crystal hydrates and anhydrous salts may be significantly different. Thus, cupric sulfate (pentahydrate) forms beautiful and transparent sky blue crystals, whereas anhydrous $CuSO_4$ is a white dust.

Usually, crystal hydrates are not too stable at high temperature, so they may be converted to anhydrous salts by heating in an oven. That is a standard test that should be done to check if some unknown salt is a crystal hydrate or an anhydrous compound.

10 points:

I cannot guarantee that this story was true, but rumor has it that among the papers left over from Sir Conan Doyle there was a draft of one story tentatively titled “Strange Crystal”. This story tells of a very rare and very valuable mineral stolen from a wealthy Mr Penrose, who earned his capital collecting and selling various unusual natural objects and artifacts. One of his most valuable items was stolen from his collection: a very rare "Patagonian sapphire", a dark blue natural mineral that, according to Penrose, was found in some remote cave in Patagonia. The Patagonian sapphire was unique because of its shape: it was a prism, and its cross-section was a regular pentagon. Penrose was the only person in the world to own a Patagonian sapphire, which was the most valuable item in his collection. Unsurprisingly, Penrose insured his Patagonian sapphire at Lloyd for a fantastic £ 10 million. Shortly thereafter, the crystal was stolen from the house of Penrose. Lloyd's representative came to Holmes and asked if he could find this extremely valuable item. Hearing about the unusual form of the mineral, Holmes immediately concluded that Lloyd was not obliged to pay Penrose money and might even sue Penrose for fraud because the Patagonian Sapphire was a clear fake, and the alleged robbery was staged.

Why did Holmes come to this conclusion?

If Holmes lived in 2010, could he be equally categorical?

Solution:

The word “sapphire” is not a key here. The problem is deeper. No other crystal with pentagonal symmetry are possible. The reason is simple: the shape of a crystal depends on its structure, and the internal structure of crystals is similar to tiling: small elements, a.k.a. "unit cells" (or simplest possible motifs of crystal structure), are densely packed to form an ordered and symmetrical structure, where the "unit cell" repeats many, many times in all three dimensions. That is similar to tiles on a 2D surface, but instead of a 2D surface, a 3D space is being filled. You may cover the 2D plane with square, trigonal or hexagonal tiles without gaps, but you cannot do that with pentagons: no symmetrical pattern without gaps can form. The same works for 3D space.

Square or hexagonal tiles may form rectangular or trigonal, or hexagonal shapes, but not pentagons. Similarly, cubic or tetrahedral unit cells may form cubic or hexagonal crystals, but not pentagonal ones. That is why pentagonal crystals are not possible.

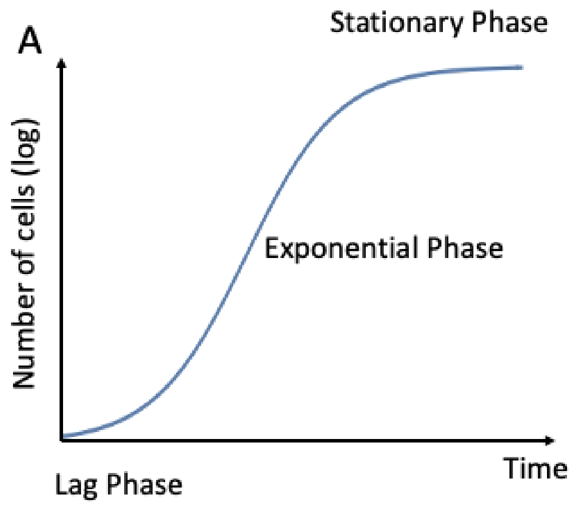
However, recently, quasicrystals were discovered. They are ordered, but their internal structure is not symmetrical: they resemble Penrose mosaic (actually, the name "Penrose" is a hint). Google it. Similarly to Penrose tiling, quasicrystals form an ordered but aperiodical structure. They can form pentagonal shapes, but Holmes couldn't know about that.They can form pentagonal "crystals", but they are not easy to prepare, and Holmes didn't know about them.

BIOLOGY

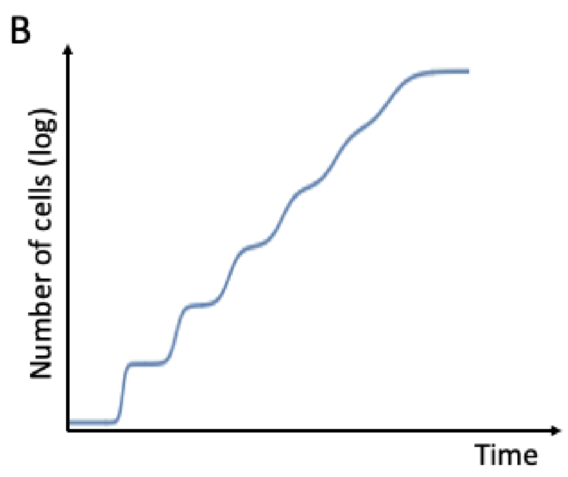
5 points:

Usually, the growth of a bacterial culture occurs as shown in the Fig A (see below): after the initial lag phase (when the growth of bacteria is slow), bacteria begin to grow exponentially (i.e., their growth rate is proportional to the density of the culture, so it doubles after the same time interval). This phase looks like a straight line in the left figure because the Y-axis is on a logarithmic scale, which means it increases as “1, 10, 100, 1000…” instead of our usual “1, 2, 3, 4…”. (This is a standard way of plotting the growth of bacterial cultures, as well as other processes where the amplitude of the change in numbers is extremely large.) As a rule, the growth of the overwhelming majority of bacterial cultures, including *E. coli*, one of the most popular bacteria in research laboratories, obeys that law.

However, if you take a small sample of the *E. coli* culture, incubate it at an elevated temperature (45 degrees), add it to a fresh media and place it under optimal temperature conditions, the growth curve will be as shown in the Fig. B. Explain why this is happening. What else can cause bacteria to grow, as shown in Figure B?



A



B

Answer:

When a single bacterion starts to grow, it divides into two new bacteria, and, after some pause, each of them divides again, and so on, and so forth. Since each of them grow in the same conditions, they divide simultaneously. If they continue to grow synchronously (i.e. divide in the same moment of time, followed by some pause, which is needed for daughter bacteria to grow and prepare for the next division), the growth curve will have clearly visible steps, whose height will be 2, 4, 8, 16, etc., in other words, it will be close to the graph B (see the above figure). However, when many bacteria start to divide at different moments of time, each of them produces their own “B-type” graphs, but they are shifted relative to each other along the time axis. As a result, the overall plot is averaged, and no steps are visible on it (the graph A).

However, imagine that we took a relatively small number of bacteria and kept them in the conditions that prevent their division. One possible way to do that is to keep it at elevated temperature, which does not kill bacteria, but their division is stopped. By doing that, we *synchronize* bacteria, which means if we put them in normal conditions, they will start to divide synchronously, i.e., each bacterion in this culture, as well as their daughter cells, divide at exactly the same moment of time. As a result, the plot will look like the B-type plot. This plot is typical for synchronized cultures. Furthermore, due to inhomogeneity of conditions, some bacteria divide with different rates, so the bacterial culture is gradually becoming desynchronized, and steps are becoming more and more blurred.

10 points:

During the 19th and first half of the 20th century, scientists believed that all species belong to two domains of life: eukaryotes and prokaryotes (a. k. a. bacteria). Recently, scientists have found that one group of unicellular organisms, despite their visible similarity to bacteria, represent a completely different domain of life, which is now called "archaea".

Archaea are very unusual organisms that look like bacteria, but differ from them ... in almost everything. Their proteins are completely different (and they are more like our own, eukaryotic proteins), their cell membrane is different, their genomic organization is different, their metabolism is different. Archaea can be found everywhere, even in such unfriendly places as Antarctic lakes, Yellowstone geysers, underwater volcanoes, and even nuclear reactors. Some of them live in our guts (and they are responsible for producing methane) and similar places. However, until now, no archaea species has been identified as the primary cause of any human disease.

Propose possible explanations for that phenomenon.

Answer:

In fact, several theories have been proposed, but there is no universal answer. The most probable explanations are based on two considerations: first, archaea are too different from bacteria and from our own cells, second, they feel more comfortable in more exotic and severe conditions, so our body is not the best place for them. Indeed, many archaea live in anaerobic conditions (so oxygen is a poison for them), or at high temperature, or high salinity, etc. Even the archaea that live in our guts are responsible for methanogenesis (an odd process where methane is formed as a product of catabolism). Clearly, our body would not be a good place for most archaea. Second, since their biochemistry and metabolism are that different, they use a different set of cofactors (small molecules that, like vitamins, serve as regulators on many processes), which make their life in our body more problematic (in contrast to bacteria). Lastly, they have totally different viruses, which prevents gene transfer between them and our cells, so they cannot quickly evolve to obtain the traits that are needed for efficient survival in our body.
For one of these (and possibly some other reasons), they prefer to live in geysers, saline lakes, and other unpleasant places rather than in our body.

LINGUISTICS

5 points:

The phrases below are from the language of an Indigenous People with the translation in a random order.

ava hoveny iyuuk
ahaaly uwaak
avaaly uwaak
masahay ahvay aym
masahay ahvay hoveny ičook

I am in the house
I gave a dress to the girl
He is by the water
I saw that house
I made that dress for the girl

1. Match the sentence to the correct translation. Explain your reasoning.
2. Translate: *He made that house for the girl*. Explain your reasoning.

Answer:

ava hoveny iyuuk -- I saw that house
ahaaly uwaak -- He is by the water
avaaly uwaak -- I am in the house
masahay ahvay aym -- I gave a dress to the girl
masahay ahvay hoveny ičook -- I made that dress for the girl

He made that house for the girl -- masahay ava hoveny ičook

Solution:

This language is Mojave, spoken by the Mojave Tribe along the Colorado River.
Uwaak, masahay, ahvay, and *hoveny* appear unchanged twice in the same location in the sentence. *Dress, girl, house, to be*, and *that* also appear twice. *Masahay* and *ahvay* appear together in two sentences, meaning they must be *girl* and *dress*. *Hoveny* must be *that* because it appears in another sentence. It is reasonable to assume that *hoveny* modifies *ahvay* based on word order, so *ahvay* must be *dress*. Now we know that *house* must be *ava* because that is the other sentence with *hoveny*. Finally, *ava* is identifiable in the other sentence with house.

To translate, we must take what we know from the given sentences. We have discovered that the basic sentence structure is indirect object -- direct object -- verb (with implied subject). So *masahay*, *girl,* must be the first word. The direct object, *ava,* and its modifier, *hovney*, must follow. Finally, the verb. We are given *ičook*, however, it is translated as first person and we need third person. Looking at *uwaak*, there is no difference between to be in the two sentences it is in, even though it is translated as third person and first person. Therefore, no modification is needed. Likewise, tense does not matter.

The sentence structure is Indirect object -- direct object -- verb (with implied subject). -ly is the locative; -m and -k verb endings may signify either past or present and either first person or third person.

10 points:

Below are some phrases from Cartínese — a language that will evolve several hundred years in the future on the islands of Cartí in Panama. Translations are presented alongside each sentence, in scrambled order.

(For phonetic reading: c = /ts/, q = /tʃ/, x = /ʃ/)

A. cui ye lulo na be qewana qe ho moha	1. he doesn't like himself
B. cui qisin dei ese a ho qewanay	2. our dogs play among themselves
C. we a qe be suruy	3. we didn't see the doctor!
D. ye qisin dei cui we be lemuy so	4. she's going to set me up
E. we ye qisin dei a be xanay	5. your dog likes to play
F. cui we ye qewan la be rakay ra	6. the doctor said that (his patient) is sick
G. ye lemo dei ye su ho xana	7. they call each other dogs
H. cui ye lekera la qe be xanaha so	8. she sees her dog
I. ye lulo qe ho xanay koi	9. you all get called by each other
J. ese su ho surun koi	10. we don't have to play ball
K. cui su a ho bodanan koi	11. i might call my lawyer
L. wocasa dei a xume be bodana ra	12. you are all likeable
M. cui we a be bodanay ka qisin	13. i see your ball
N. we ye be rakay ra	14. the ball isn't visible
O. qisin dei ye su ho suruy ka qewana	15. they will set up the game!
P. lekera mu ye we be burunay ho danay so	16. their dog bit them

1. Match each sentence to the right to a sentence to the left.
2. One sentence on the left is written incorrectly; rewrite the sentence to reflect the meaning of the sentence to the right, and explain why.
3. Translate the following sentences into Cartínese:
   a. I like that you aren't sick.
   b. You all have eyes!
   c. She will call herself.
   d. They are set-uppable.

Answer:

A10	B2	C1	D16
E8	F15	G4	H3
I14	J12	K9	L11
M7	N13	O5	P6

A. cui ye lulo na be qewana qe ho moha	10. we don't have to play ball
B. cui qisin dei ese a ho qewanay	2. our dogs play among themselves
C. we a qe be suruy	1. he doesn't like himself

D. ye qisin dei cui we be lemoy so	**16. their dog bit them**
E. we ye qisin dei a be lekeray	8. she heals her dog
F. cui we ye qewan la be rakuy ra	15. they will set up the game!
G. ye lemo dei ye su ho xana	13. i see your teeth
H. cui ye lekera la qe be xanaha so	3. we didn’t see the doctor!
I. ye lulo qe ho xanay koi	14. the ball isn’t visible
J. ese su ho surun koi	12. you are all likeable
K. cui su a ho bodanan koi	9. you all get called by each other
L. wocasa dei a xume be bodana ra	11. i might call my lawyer
M. cui we a be bodanay ka qisin	7. they call each other dogs
N. we ye be rakuy ra	4. she’s going to set me up
O. qisin dei ye su ho suruy ka qewana	5. your dog likes to play
P. lekera mu ye we be burunay ho danay so	6. the doctor said that (his patient) is sick

Correct translation: **()** qisin dei **ese** ce be lemuy so

a. I like that you aren’t sick. **mu ye su qe be burunay ho suru**
b. You all have dogs! **cui su ese xana la be mohan so**
c. She will call herself. **ce a be bodanay ra**
d. They are set-uppable. **ese we ho rakuy koi**

Solution:

(ergative case for intransitive verbs, absolutive case for transitive verbs)
Articles (D): **ye** = singular object article; **ese** / **cui** = plural subject / object article
Verb prefixes (P): **be** = absolutive verb prefix, **ho** = ergative verb prefix
a = reflexive pronoun, **dei** = possessive, **qe** = negation, **la** = exclamation, **xume** - modal
su = 2p pronoun, **we** = 3p pronoun   Tense (T): **so** = past tense, **ra** = future tense

bodana = call, **lemo** = bite (teeth), **lekera** = heal (doctor), **lulo** = ball, **moha** = have, **raku** = set up, **suru** = to like, **qewana** = play (game), **qisin** = dog(s), **wocasa** = lawyer, **xana** = see

OP → ye/ese (N) OR **a**   TP → P N VP (SOV word order)
VP → (**mu** + TP) (OP) (**la, xume**) (**qe**) P V (T); conjugation: 1p: **-**∅, 2p: **-n**, 3p: **-y**
Passive phrases: OP **ho** V **koi**, where **koi** marks adjectives

COMPUTER SCIENCE

- Your program should be written in Java or Python-3
- No GUI should be used in your program: eg., easygui in Python
- All the input and output should be via files named as specified in the problem statement
- Java programs should be submitted in a file with extension .java; Python-3 programs should be submitted in a file with extension .py
- **No .pdf, .doc, .docx, etc! Programs submitted in incorrect format will not receive any points!**

5 points:

There are *n* building projects currently proposed at sites arranged in a row along the main commercial street in Elizaville, NY. Each one, if building there were allowed, would bring a specific amount of revenue to the town: site 1 would bring revenue r_1 (in dollars), site 2 would bring revenue r_2, etc.

The town would like to approve *three* of these projects to bring in as much revenue as possible. However, there is a density constraint: no two approved projects can be next to each other! So their goal is to pick three projects $i,\ j,\ k$ which maximize the sum of the revenues $r_i + r_j + r_k$, but where $j \geq i + 2$ and $k \geq j + 2$.

Write a program that takes as input the revenues $r_1,\ldots, r_n$ (in a file named **input.txt**, formatted as one integer per line) and outputs the list of projects to approve (into an output file named **output.txt**, also one integer per line).

For example, if input.txt contains:

```
8
10
12
11
2
4
5
```

you would select projects 2, 4, 7, for a total revenue of 10 + 11 + 5 = 26, and the output.txt file would contain:

```
2
4
7
```

Your program should run in a few seconds when the input has length up to *n* = 50. x

Solution:

Python-3:

```
import sys
import numpy as np
import os

def parseInput(filename = None):
    if filename == None:
        weights = readIntList(sys.stdin)
    else:
        with open(filename, 'r') as f:
            weights = readIntList(f)
    return weights

def readIntList(f):
    #To do: add error checks for parsing student output
    ints_list = []
    for line in f.readlines():
        if line != '\n':
            ints_list.append(int(line))
    return ints_list

def writeOutput(optset, filename = None):
    if filename == None:
        writeIntList(optset, sys.stdout)
    else:
        with open(filename, 'w') as f:
            writeIntList(optset, f)

def writeIntList(optset, f):
    for x in optset:
        f.write(str(x) + os.linesep)
    return

def constrainedWISL(weights, k):
    # 5-pointer solutuion (with k=3)
    # weights is a list of nonegative weights
    # k is a positive integer
    n = len(weights)
    opt = np.zeros((n, k))
    # opt[i,j] will store the value of heaviest IS of size at most j+1 among
nodes 0,...,i.
    for j in range(k):
        opt[0,j] = weights[0]
        opt[1,j] = max(weights[0],weights[1])
        for i in range(2,n):
            if j == 0:
                opt[i,j] = max(opt[i-1,j], weights[i])
            else:
                opt[i,j] = max(opt[i-1,j], opt[i-2,j-1] + weights[i])
    # Now find optimal set
    i=n-1
    j=k-1
    optset = []
    while i>0 and j>=0:
        if opt[i,j] == opt[i-1,j]:
            i = i-1
        else:
            optset.append(i)
            i = i-2
            j = j-1
    if j>=0 and i==0:
        optset.append(0)
    optset.reverse()
    ### Now add 1 to all indices to correct for the numbering from 1.
    adjusted_optset = [x+1 for x in optset]
    return adjusted_optset

def main():
    weights = parseInput("input.txt") # Reads from input.txt file
    optset5 = constrainedWISL(weights, k=3)
    writeOutput(optset5, "output.txt") #  prints output to output.txt
    return

if __name__=="__main__":
    main()
```

10 points:

You were so successful in helping out Elizaville, NY that you've been asked to help out a much larger city—Los Angeles, CA—with a similar problem. Again, there are *n* sites arranged along a street, Figueroa Boulevard. Figueroa is about 30 miles long and has space for lots of building sites, so *n* could be huge.

This time, however, there is no limit on the number of sites that may be selected. The only constraint is that you may not select two sites that are next to each other.

Write a program that takes as input the revenues $r_1, \ldots, r_n$ (in a file named **input.txt**, formatted as one integer per line) and outputs the list of projects to approve (into an output file named **output.txt**, also one integer per line).

For example, if input.txt contains (same as above):

```
8
10
12
11
2
4
5
```

you would select projects 1, 3, 5, and 7, for a total revenue of 8 + 12 + 2 + 5 = 27, and the output.txt file would contain:

```
1
3
5
7
```

For this problem, your program should be able to handle larger inputs—with *n* as large as 10,000—in under a second. We recommend you design an algorithm that solves the problem in time roughly proportional to *n*.

Solution:

Python-3:

```
import sys
import numpy as np
import os


def parseInput(filename = None):
    if filename == None:
        weights = readIntList(sys.stdin)
    else:
        with open(filename, 'r') as f:
            weights = readIntList(f)
    return weights

def readIntList(f):
    #To do: add error checks for parsing student output
    ints_list = []
    for line in f.readlines():
        if line != '\n':
            ints_list.append(int(line))
    return ints_list

def writeOutput(optset, filename = None):
    if filename == None:
        writeIntList(optset, sys.stdout)
    else:
        with open(filename, 'w') as f:
            writeIntList(optset, f)

def writeIntList(optset, f):
    for x in optset:
        f.write(str(x) + os.linesep)
    return

def WISL(weights):
    # 10-pointer solution
    # weights is a nonempty list of nonengative real numbers
    # weights[i] is the weight of vertex i
    n = len(weights)
    # Fill the table:
    # opt[i] is the weight of the heaviest independent set
```

```python
    # among vertices 0,1,...,i.
    opt = np.zeros(n)
    opt[0] = weights[0]
    opt[1] = max(weights[0],weights[1])
    for i in range(2,n):
        opt[i] = max(opt[i-1], opt[i-2] + weights[i])
    #Now compute the optimal set
    optset = []
    i = n-1
    while i>0:
        if opt[i] == opt[i-1]:
            i = i-1
        else:
            optset.append(i)
            i = i-2
    if i==0:
        optset.append(0)
    optset.reverse()
    ### Now add 1 to all indices to correct for the numbering from 1.
    adjusted_optset = [x+1 for x in optset]
    return adjusted_optset

def main():
    weights = parseInput(“input.txt”) # Reads from input.txt
    optset10 = WISL(weights)
    writeOutput(optset10, “output.txt”) #  prints output to output.txt
    return

if __name__=="__main__":
    main()
```

Java:

Solution courtesy of Boyan Litchev

```java
import java.util.Scanner;
import java.io.PrintWriter;
import java.io.File;
import java.util.ArrayList;


class tenPointsOctober {
  public static void main(String[] args) throws Exception {
    Scanner in = new Scanner(new File("input.txt"));

    // these sets store all projects picked so far
    ArrayList<Integer> prevSet = new ArrayList<>();
    ArrayList<Integer> prevPrevSet = new ArrayList<>();
    ArrayList<Integer> temp;
    long prevMax = 0;
    long prevPrevMax = 0;
    long currMax = 0;
    int curr;
    int projectNumTracker = 1;
    while(in.hasNextInt()){
      curr = in.nextInt();
      if(prevMax >= prevPrevMax + curr){
        prevPrevSet = (ArrayList<Integer>)prevSet.clone();
        currMax = prevMax;
      }else{
        temp = prevSet;
        prevPrevSet.add(projectNumTracker);
        prevSet = prevPrevSet;
        prevPrevSet = temp;
        currMax = prevPrevMax + curr;
      }
      prevPrevMax = prevMax;
      prevMax = currMax;
      projectNumTracker ++;
    }

    PrintWriter out = new PrintWriter("output.txt");
    for(int i = 0; i< prevSet.size(); i++){
      out.println(prevSet.get(i));
    }
    out.close();
  }
}
```