Question 1

[**7 marks**] After watching the movie WarGames, Bob decides to quit his job in business consulting to dabble in cyber security instead. Unfortunately, Bob knows nothing about writing exploits but is familiar with the amazing resource that is StackOverflow. Here, he manages to narrow down n different cyber security related topics, where each topic i has l_i lines of exploit code. The exploit code related to the i^{th} topic also has a "hackerscore" of h_i per line and a computational cost of c_i per line.

Bob believes that if he merges x and y lines of exploit code from topics i and j respectively, he creates a new exploit with "hackerscore" $h_i x + h_j y$ and computational cost $c_i x + c_j y$. He can continue to merge this new exploit with code from other topics until he is satisfied.

Obviously, Bob wants to either go big or go back to consulting. To hack into the NSA, he needs to create an exploit with "hackerscore" exactly H, but it must also run on his old corporate laptop with computational power C. Help Bob find the minimum number of lines of code he'll need to copy and paste to create a valid exploit.

1.1 [**2 marks**] Bob isn't sure whether to formulate this problem with Linear Programming or Integer Linear Programming. Decide which one to use and justify your choice.

Integer Linear Programming. The variables, which is the number of lines of a topic are integers.

1.2 [**1 mark**] List the variables.

Let n be the total number of topics. The variables are integers
k_i, $1 \leq i \leq n$,which is the number of lines copy and pasted for topic i

1.3 [**3 marks**] List the constraints, and justify each one.

Let n be the total number of topics.

[A]

$$\sum_{i=1}^{n} c_i k_i \leq C$$

Total computational cost must be less than or equal to the computational power C.

[B]

$$\sum_{i=1}^{n} h_i k_i = H$$

Sum of "hackerscore" is exactly H.

[C]

$$0 \leq k_i$$

and

$$k_i \leq l_i$$

The number of lines chosen for each topic must be larger than or equal to 0 and less than or equal to the total number of lines for that topic.

1.4 [**1 mark**] Provide the objective function.

Let n be the total number of topics. Minimize

$$\sum_{i=1}^{n} k_i$$

Question 2 *Treasure Rooms*

[**12 marks**] You are inside a 3D cube building with $m \times m \times m$ rooms (i.e it has m floors, where each floor has m rows and m columns of rooms). Starting at any room, you can travel to $n < m$ other rooms where at any room, you can only travel up or down, left or right, backwards or forwards to the next room in the building. You cannot travel diagonally between rooms, and you also cannot revisit any room.

You are given the location of k different rooms $t_1, t_2, ..., t_k$ in the building, each of which contains a treasure. Your goal is to travel to as many of these k treasure rooms as possible within your path of length n, and you wish to use Linear Programming or Integer Linear Programming to find an optimal path.

2.1 [**2 marks**] You aren't sure whether to formulate this problem with Linear Programming or Integer Linear Programming. Decide which one to use and justify your choice.

Integer Linear Programming. The variables indicate if a room in include in the path, this can is either true or false and can be represented by 0 and 1.

2.2 [**2 marks**] List the variables.

Let i, j, k indicate a room in a building, m indicate it is the pth room in a path. The variables are x_{ijkp} for $1 \leq i \leq m, 1 \leq j \leq m, 1 \leq k \leq m, 1 \leq p \leq n$

2.3 [**6 marks**] List the constraints, and justify each one.

Note: To achieve full marks in Question 2.2 and Question 2.3 you need to provide a formulation which uses $\Theta(nm)$ variables. Half marks will be awarded for a formulation which uses a number of variables and constraints polynomial in n, m, and k (the template solution uses $\Theta(nm^3)$).

Abdallah believes finding an encoding in $\Theta(nm)$ variables can be tricky and students are encouraged to manage their time wisely. For instance the following approach sounds reasonable: find any polynomial encoding for Q2, solve the rest of the assignment, then come back to Q2 to try and find a $\Theta(nm)$ encoding

[A]

$$0 \leq x_{ijkp} \leq 1 \text{ for } 1 \leq i \leq m, 1 \leq j \leq m, 1 \leq k \leq m, 1 \leq p \leq n$$

x_{ijkp} is either 0 or 1 as a room is either included or not included in a path.

[B]

$$0 \leq \sum_{p=1}^{n} x_{ijkp} \leq 1 \text{ for } 1 \leq i \leq m, 1 \leq j \leq m, 1 \leq k \leq m$$

For any room i, j, k, the number of times it can be included in a path is either 1 or 0.

[C]

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} x_{ijkp} = 1 \text{ for } 1 \leq p \leq n$$

Exactly one room can be the pth room of a path.

[D]

$$x_{ijkp} - (x_{(i-1)jk(p-1)} + x_{(i+1)jk(p-1)} + x_{i(j-1)k(p-1)} + x_{i(j+1)k(p-1)} + x_{ij(k-1)(p-1)} + x_{ij(k+1)(p-1)}) \leq 0$$

for $1 \leq i \leq m, 1 \leq j \leq m, 1 \leq k \leq m, 1 \leq p \leq n$.
If the room (i, j, k) is not the pth room of a path, then $x_{ijkp} = 0$ and the inequality will be

satisfied. If the room (i, j, k) is the pth room of a path, then $x_{ijkp} = 1$ and the inequality is only satisfied when the summation term in the brackets is larger than or equal to 1, and this is only true when one of the neighbouring rooms is the $p-1$th room in the path. However, we need to add a constraint so that the inequality works for the first room in a path. The additional constraint is in below

[E]

$$x_{ijk0} = 1 \text{ for } 1 \le i \le m, 1 \le j \le m, 1 \le k \le m$$

2.4 **[1 mark]** Provide the objective function.

Let $t_{ijk} = 1$ if a treasure exist in the room (i, j, k), 0 otherwise. We need to maximize

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} t_{ijk} \sum_{p=1}^{n} x_{ijkp}$$

2.5 **[1 mark]** Can a solution be derived in polynomial time?

No. Integer Linear Programming is NP-complete.

Question 3 *Subset Sum*

[8 marks] Consider the following problem.

Subset Sum:
Instance: An integer array $A[1, \dots, n]$.
Question: Is there a subset $I \subseteq \{1, \dots, n\}$ such that

$$\sum_{i \in I} A[i] = \sum_{i \notin I} A[i]?$$

3.1 **[4 marks]** We want to build a dynamic programming algorithm to answer Subset Sum. Define the subproblems, the base cases and the recurrence relation between the subproblems, and explain the recurrence relation in words.

The problem is equivalent to
Question: Is there a subset $I \subseteq \{1, \dots, n\}$ such that

$$\sum_{i \in I} A[i] = \frac{1}{2}\sum_{i=1}^{n} A[i]?$$

Define the sub-problem $M(i, j)$ as does there exist $I \subseteq \{1, \dots, i\}$ such that $\sum_{i \in I} A[i] = j$. The solution to this sub-problem is $m(i, j)$, which is either 1(True) or 0(False). The base cases are

$$m(i, 0) = 1 \text{ for } 0 \le i \le |A|$$

$$m(0, j) = 0 \text{ for } 1 \le j \le \frac{1}{2}\sum_{i=1}^{n} A[i]$$

Every set contains an empty set which sums up to 0, and an empty set can never have subset that sums up to a value larger than 0.
The recurrence relation is

$$m(i, j) = max(m(i-1, j), m(i-1, j - A[i])) \text{ for } 1 \le i \le |A|, 1 \le j \le \frac{1}{2}\sum_{i=1}^{n} A[i]$$

Subset $I_1 \in 1,\ldots,i$ has subset sum j either if there exist $I_2 \in 1,\ldots,i-1$ that sums up to j or there exist $I_3 \in 1,\ldots,i-1$ that sums up to $j - A[i]$. The *max* function returns 1 is at least one of the two cases is 1, otherwise it returns 0. The solution for out of bound cases is 0.

3.2 **[2 marks]** What would be the worst-case time complexity of an algorithm based on your approach? Justify your answer. There are $|A| * \frac{1}{2}\sum_{i=1}^{n} A[i]$ sub-problems, computing each requires $O(1)$. The time complexity $O(nT)$, where $n = |A|$ and T is the total sum of array A.

3.3 **[2 marks]** Does the existence of this algorithm prove that Subset Sum is in class P? Justify your answer.

No. $O(nT)$ is the pseudo-polynomial time. T is the input magnitude and the linear growth of the number of input bits has a exponential growth in T, so the time complexity is exponential in terms of input length and so is not in class P

Question 4 *Circuit Differentiation*

[14 marks] You are working for CircuitZ, a company which designs and manufactures circuits for use in industrial equipment. You have recently discovered a revolutionary method of circuit design, which requires you to use only NAND gates in your circuits. A NAND gate works on two boolean inputs and has one boolean output. In particular, if x_1 and x_2 are inputs, and y is the output, then the truth table is:

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

In propositional logic, this can be written as $y \equiv \overline{x_1 \wedge x_2}$, or more simply as $y \equiv \text{NAND}(x_1, x_2)$. A circuit can then be expressed as a list of NAND gates, which may involve other NAND gate outputs as their inputs, along with a defined output variable.

4.1 **[1 mark]** Given the inputs $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, calculate the output variable y_1 for the circuit below.

$$y_1 \equiv \text{NAND}(x_1, y_3), \qquad y_2 \equiv \text{NAND}(x_2, x_3), \qquad y_3 \equiv \text{NAND}(x_1, y_2).$$

$y_1 = 1$

4.2 **[3 marks]** You have many circuits already designed using other types of gates, and want to convert these into circuits using only NAND gates. Show how the following gates can be written as circuits involving only NAND gates.

[A] The NOT gate $y \equiv \text{NOT}(x)$, which inverts its input.

[B] The AND gate $y \equiv \text{AND}(x_1, x_2)$, which is 1 if both inputs are 1.

[C] The OR gate $y \equiv \text{OR}(x_1, x_2)$, which is 1 if any input is 1.

$\text{NOT}(x) = \text{NAND}(x, 1)$
$\text{AND}(x_1, x_2) = \text{NOT}(\text{NAND}(x_1, x_2)) = \text{NAND}(\text{NAND}(x_1, x_2), 1)$
$\text{OR}(x_1, x_2) = \text{NOT}(\text{AND}(x_1, x_2)) = \text{NAND}(\text{NAND}(\text{NAND}(x_1, x_2)), 1), 1)$

With your new design method set up, you have decided to convert all of your circuits to this new system. You have asked your employees to do the conversion, but they don't need to use the method of Question 4.2. Since mistakes might happen, you have given each circuit to two employees, and want to compare their answers to check if it's correct. Of course, there are multiple ways to do the conversion, so you can't easily check whether the two circuits are the same. Consider the following decision problem.

CircDiff:
Instance: Two circuits C_1 and C_2, each with n inputs $x_1, \dots, x_n$ and one output, and a list of NAND gates.
Question: Does C_1 compute a different function than C_2?

In other words, you want to find out whether two two circuits your employees have created are different for some combination of inputs.

You may assume that circuits will never have loops (in the directed sense). That is, no gate output will affect its input (by looping back around, for example).

4.3 [3 marks] Show that CircDiff is in the class NP.

Assume the output for each circuit is computed in polynomial time. The algorithm S which takes $C_1, C_2, x_1, \dots, x_n$, where $x_1, \dots, x_n$ is the certificate for CircDiff, and solves the problem $C_1(x_1, \dots, x_n) \neq C_(x_1, \dots, x_n)$. If CirDiff returns true, then there is at least one combination of input that generate different output for C_1 and C_2, which means there exist $x_1, \dots, x_n$ such that $S(C_1, C_2, x_1, \dots, x_n)$ returns true, and S runs in polynomial time, hence CircDiff is in class NP.

4.4 [7 marks] By reducing from 3-SAT, show that CircDiff is NP-hard.

As shown in 4.2, NOT, AND and OR gate can simply be represented by multiple NAND gates. Thus the following solution will simply write boolean expression in terms of $\wedge, \vee$ and $\neg$.

For any 3-CNF ϕ with input $x_1, \dots, x_n$, construct two boolean functions

$$p_1(x_1, \dots, x_n) = \phi(x_1, \dots, x_n) \wedge 0$$
$$p_2(x_1, \dots, x_n) = \phi(x_1, \dots, x_n) \wedge 1$$

Let C_1 and C_2 be the NAND logic gate version of p_1 and p_2. This conversion is in polynomial time. For 3-CNF with n literals, Each circuit will have $\frac{n}{3} + 1$ AND gates, and $\frac{2n}{3} + 2$ OR gates. Conversion of a single OR or AND gate to an NAND gate is in polynomial time, so converting all gates to NAND in a 3-CNF expression is done in polynomial time.

If 3-SAT(ϕ) is true, then ϕ is satisfiable and there exist a combination of $x_1, \dots, x_n$ such that $phi(x_1, \dots, x_n) = 1$, then that same input will make $p_1 = 0$ and $p_2 = 1$, which means p_1 and p_2 are not the same expression. Since C_1 is equivalent to p_1 and C_2 is equivalent to p_2, C_1 computes a different function than C_2 and CircDiff(C_1, C_2) is true

If CircDiff(C_1, C_2) is true, then C_1 and C_2 are different functions and then there exist some input $x_1, \dots, x_n$ where $C_1(x_1, \dots, x_n) \neq C_2(x_1, \dots, x_n)$. But this means that $C_2(x_1, \dots, x_n) = 1$ as $C_1 = 0$ for any possible inputs. Since C_2 and p_2 are just equivalent functions, $p_1(x_1, \dots, x_n) = 1$, and this is only possible if $\phi(x_1, \dots, x_n) = 1$. Hence 3-SAT($\phi$) is true.

3-SAT is np-hard, and since 3-SAT can be reduced to CircDiff in polynomial time, all problems in NP can be reduced to CircDiff, so CircDiff is NP-hard as well.