- Homework solutions should be neatly written or typed and turned in through **Gradescope** by 11:59 pm on the due date. No late homework will be accepted for any reason. You will be able to look at your scanned work before submitting it. Please ensure that your submission is legible (neatly written and not too faint), or your homework may not be graded.
- Students should consult their textbook, class notes, lecture slides, instructor, and TAs when they need help with homework. Students should not look for answers to homework problems in other texts or sources, including the internet. Only post about graded homework questions on Piazza if you suspect a typo in the assignment or if you don't understand what the question is asking you to do.
- Your assignments in this class will be evaluated not only on the correctness of your answers but on your ability to present your ideas clearly and logically. You should always explain how you arrived at your conclusions using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.
- For questions requiring pseudocode, you can follow the same format as we do in class or write pseudocode in your style, as long as you specify your notation. For example, are you using "=" to mean assignment or to check equality? You are welcome to use any algorithm from class as a subroutine in your pseudocode. For example, if you want to sort list A using `InsertionSort`, you can call `InsertionSort(A)` instead of writing out the pseudocode for `InsertionSort`.

Name : Ganraj A. Borade
Entry number : 2023MCS2478

There are 6 questions for a total of 100 points.

---

1. (*Analysing recursive functions*) In this question, you are asked to analyse two recursive functions.
   (a) (10 points) Consider the following recursive function that takes as input a positive integer.

```
F($n$)
    · If ($n > 1$) F($\lfloor n/2 \rfloor$)
    · print("Hello World")
```

Give the **exact** expression, in terms of n, for the number of times "Hello World" is printed when a call to $F(n)$ is made. Argue the correctness of your expression using mathematical induction.

[**Solution**]

Let's take $T(n)$ denotes the function which denotes - number of times "Hello World" is printed when $n >= 1$ input is provided to the function $F(n)$.

By the recurrence relation, we can say that $T(n) = T(\lfloor n/2 \rfloor) + 1$.

Now let's consider the cases one by one (mentioned in Box 1):

Box 1

1. $T(1) = 1$ (we don't require the recurrence relation for this. We can clearly see this from the pseudo code mentioned in the question).

Box 1

2. $T(2) = T\left(\left\lfloor\frac{2}{2}\right\rfloor\right) + 1 = T(1) + 1 = 1 + 1 = 2$ (from point 1).
3. $T(3) = T\left(\left\lfloor\frac{3}{2}\right\rfloor\right) + 1 = T(1) + 1 = 2$.
4. $T(4) = T\left(\left\lfloor\frac{4}{2}\right\rfloor\right) + 1 = T(2) + 1 = 2 + 1 = 3$ (from point 2).
5. $T(5) = T\left(\left\lfloor\frac{5}{2}\right\rfloor\right) + 1 = T(2) + 1 = 3$.
6. $T(6) = T\left(\left\lfloor\frac{6}{2}\right\rfloor\right) + 1 = T(3) + 1 = 3$ (from point 3).
7. $T(7) = T\left(\left\lfloor\frac{7}{2}\right\rfloor\right) + 1 = T(3) + 1 = 3$.
8. $T(8) = T\left(\left\lfloor\frac{8}{2}\right\rfloor\right) + 1 = T(4) + 1 = 3 + 1 = 4$ (from point 4).
9. $T(9) = T\left(\left\lfloor\frac{9}{2}\right\rfloor\right) + 1 = T(4) + 1 = 4$.
10. $T(10) = T\left(\left\lfloor\frac{10}{2}\right\rfloor\right) + 1 = T(5) + 1 = 4$ (from point 5).
11. $T(11) = T\left(\left\lfloor\frac{11}{2}\right\rfloor\right) + 1 = T(5) + 1 = 4$.
12. $T(12) = T\left(\left\lfloor\frac{12}{2}\right\rfloor\right) + 1 = T(6) + 1 = 4$ (from point 6).
13. $T(13) = T\left(\left\lfloor\frac{13}{2}\right\rfloor\right) + 1 = T(6) + 1 = 4$.
14. $T(14) = T\left(\left\lfloor\frac{14}{2}\right\rfloor\right) + 1 = T(7) + 1 = 4$ (from point 7).
15. $T(15) = T\left(\left\lfloor\frac{15}{2}\right\rfloor\right) + 1 = T(7) + 1 = 4$.
16. $T(16) = T\left(\left\lfloor\frac{16}{2}\right\rfloor\right) + 1 = T(8) + 1 = 4 + 1 = 5$ (from point 8).

   $\vdots$

   $\vdots$

30. $T(30) = T\left(\left\lfloor\frac{30}{2}\right\rfloor\right) + 1 = T(15) + 1 = 4 + 1 = 5$ (from point 15).
31. $T(31) = T\left(\left\lfloor\frac{31}{2}\right\rfloor\right) + 1 = T(15) + 1 = 5$.
32. $T(32) = T\left(\left\lfloor\frac{32}{2}\right\rfloor\right) + 1 = T(16) + 1 = 5 + 1 = 6$ (from point 16).

   $\vdots$

   $\vdots$

Hence, from the above points and equations, we can derive the relation easily:

$$\boxed{\text{Whenever } 2^k \leq n < 2^{k+1}, \text{then } T(n) = k + 1}$$

Here k is an integer starting from 0 and moving in the positive direction on the integer line. Let's just quickly verify the above expression (in Box 2):

Box 2

- Take $k = 0 \implies 1 \leq n < 2$ (as n can only be an integer). Hence, $T(1) = 0 + 1 = 1$.
- Take $k = 1 \implies 2 \leq n < 4$, i.e., there are 2 cases possible here:
  - $n = 2 \implies T(2) = k + 1 = 1 + 1 = 2$.
  - $n = 3 \implies T(3) = k + 1 = 1 + 1 = 2$.
- Take $k = 2 \implies 4 \leq n < 8$, i.e., there are 4 cases possible here:
  - $n = 4 \implies T(4) = k + 1 = 2 + 1 = 3$.
  - $n = 5 \implies T(5) = k + 1 = 2 + 1 = 3$.
  - $n = 6 \implies T(6) = k + 1 = 2 + 1 = 3$.
  - $n = 7 \implies T(7) = k + 1 = 2 + 1 = 3$.
- Take $k = 3 \implies 8 \leq n < 16$, i.e., there are 8 cases possible here:
  - $n = 8 \implies T(8) = k + 1 = 3 + 1 = 4$.
  - $n = 9 \implies T(9) = k + 1 = 3 + 1 = 4$.
  - $n = 10 \implies T(10) = k + 1 = 3 + 1 = 4$.
  - $n = 11 \implies T(11) = k + 1 = 3 + 1 = 4$.
  - $n = 12 \implies T(12) = k + 1 = 3 + 1 = 4$.
  - $n = 13 \implies T(13) = k + 1 = 3 + 1 = 4$.
  - $n = 14 \implies T(14) = k + 1 = 3 + 1 = 4$.
  - $n = 15 \implies T(15) = k + 1 = 3 + 1 = 4$.

$\vdots$

$\vdots$

Hence, we are getting the right results by seeing Box 1 and Box 2.
Therefore, $T(n) = k + 1$ whenever $2^k \leq n < 2^{k+1}$, where k is an integer starting from 0 and moving in the positive direction on the integer line.

$$\boxed{\begin{array}{c} T(n) = k + 1 \text{ whenever } 2^k \leq n < 2^{k+1}, \\ k \text{ is an integer starting from 0 and moving in the positive direction on the integer line.} \end{array}}$$

Let's write this expression purely in terms of n:

$$2^k \leq n < 2^{k+1}.$$

Taking the logarithm (base 2) on both sides:

$$k \leq \log n < k + 1 \quad \text{(as } \log \text{ is an increasing function).}$$

Now from the above relation, we can clearly write $\lfloor \log n \rfloor = k$ as $\log n$ values will always be less than $k + 1$ and greater than or equal to k.
So our final answer for the number of times "Hello World" will be printed will be equal to (in terms of n):

$$T(n) = \lfloor \log n \rfloor + 1 \quad \text{where } n \text{ is a positive integer.}$$

[Proof by Mathematical Induction]

So here our proposition is : $\boxed{\text{T(n)} = \lfloor \log n \rfloor + 1 \text{ such that } n \text{ is a positive integer.}}$

Basis step:

$T(1)$ is true. Since by the recurrence relation we can see that $T(1) = 1$ (as $n > 1$ is not satisfied in the pseudo code and "Hello World" will be printed only once).

And by the above formula, $T(1) = \lfloor \log(1) \rfloor + 1 = 0 + 1 = 1$.

Hence, the basis step is proved.

Inductive step:

Assume that $T(1), T(2), \ldots, T(k)$ are true for any arbitrary integer $k > 1$.

- We can write $T(k) = (\lfloor \log(k) \rfloor + 1)$ as we have assumed $T(k)$ to be true.
- $T(k+1) = T(\lfloor (k+1)/2 \rfloor) + 1$ (we can write this by the recurrence relation written at the start of the solution : $T(n) = T(\lfloor n/2 \rfloor) + 1$.).

  Since $1 \leq \lfloor (k+1)/2 \rfloor < k$ (as $k > 1$) and since $\lfloor (k+1)/2 \rfloor$ is an integer, we can write $T(\lfloor (k+1)/2 \rfloor) = \lfloor \log(\lfloor (k+1)/2 \rfloor) \rfloor + 1$, since we assumed $T(1), T(2), T(3), \ldots, T(k)$ to be true.

  So, $\boxed{\text{T(k + 1)} = \lfloor \log(\lfloor (k+1)/2 \rfloor) \rfloor + 1 + 1.}$
- Now, we can again take 2 possibilities for $k+1$: either it can be even or it can be odd.
  - $\boxed{k + 1 \textbf{ is an odd integer.}}$
    * So we can write $k+1$ to be $2m+1$, where m is any positive integer.
    * Hence, $\lfloor (k+1)/2 \rfloor = \lfloor (2m+1)/2 \rfloor = \lfloor m + (1/2) \rfloor$.
    * Since here m is any positive integer, therefore $\lfloor m + (1/2) \rfloor = m = k/2$ (as $k+1 = 2m+1$, hence $m = k/2$).
    * Therefore, now $T(k+1) = \lfloor \log(k/2) \rfloor + 1 + 1$, we can write it as $T(k+1) = \lfloor \log(k/2) + 1 \rfloor + 1 = \lfloor \log(k/2) + \log 2 \rfloor + 1 = \lfloor \log(k) \rfloor + 1$.
    * So finally we got, $\boxed{\text{T(k + 1)} = \lfloor \log(k) \rfloor + 1.}$
    * Now, as $k+1$ is an odd integer, so k has to be an even integer. Let's take $2^p \leq k < 2^{p+1}$ where p is any positive integer (we can assume this as k is any integer, we can vary values of p to get complete set of k for operating on problem). Taking the logarithm on both sides of this equation, we get: $p \leq \log k < p + 1$. So, $\lfloor \log k \rfloor = p$
    * Now, since $k < 2^{p+1}$ and k is an even integer, hence the maximum value of k possible is $2^{p+1} - 2$. i.e., the maximum value of $k+1$ will be $2^{p+1} - 1$. We can write $2^p < k + 1 < 2^{p+1}$ by the above discussion. Taking the logarithm on both sides: $p < \log(k+1) < p + 1$ (as logarithm is increasing function). So, $\lfloor \log(k+1) \rfloor = p$.
    * Hence $\lfloor \log(k) \rfloor = \lfloor \log(k+1) \rfloor$. furthur, $\boxed{\text{T(k + 1)} = \lfloor \log(k+1) \rfloor + 1}$. Hence we have proved that T(k+1) holds assuming $T(1), T(2), \ldots, T(k)$ are true when $k+1$ is odd.

- $\boxed{k+1 \textbf{ is an even integer.}}$
    - Hence $\lfloor (k+1)/2 \rfloor = (k+1)/2$.
    - So, $T(k+1) = \lfloor \log((k+1)/2) \rfloor + 1 + 1 = \lfloor \log((k+1)/2) + 1 \rfloor + 1 = \lfloor \log((k+1)/2) + \log 2 \rfloor + 1 = \lfloor \log(k+1) \rfloor + 1$
    - Finally, we got $\boxed{\text{T(k + 1)} = \lfloor \log(k+1) \rfloor + 1}$. Hence we have proved that T(k+1) holds assuming $T(1), T(2), \ldots, T(k)$ are true when $k+1$ is even.

(b) (15 points) Consider the following recursive function that takes as input two positive integers n and m.

```
G(n, m)
    · if (n = 0 OR m = 0) return;
    · print("Hello World")
    · G(n − 1, m)
    · G(n, m − 1)
```

Give the **exact** expression, in terms of n and m, for the number of times "Hello World" is printed when a call to G(n, m) is made. Argue the correctness of your expression using mathematical induction.

[**Solution**]

Let's take $T(n, m)$ to be the number of times "Hello World" is printed when n, m input is provided to function G, i.e., $G(n, m)$.
By the recurrence relation, we can say that $T(n, m) = 1 + T(n-1, m) + T(n, m-1)$.
Let's take some cases and understand the pattern.

- $T(1,1) = T(0,1) + T(1,0) + 1 = 0 + 0 + 1 = 1$
- $T(1,2) = T(0,2) + T(1,1) + 1 = 0 + 1 + 1 = 2$
- $T(1,3) = T(0,3) + T(1,2) + 1 = 0 + 2 + 1 = 3$

  ⋮

  ⋮
- $T(1, m-1) = T(0, m-1) + T(1, m-2) = 0 + (m-2) + 1 = m - 1$
- $T(1, m) = T(0, m) + T(1, m-1) = 0 + (m-1) + 1 = m$

- $T(2,1) = T(1,1) + T(2,0) + 1 = 1 + 1 = \binom{2}{2} + 1$
- $T(2,2) = T(1,2) + T(2,1) + 1 = 2 + (1+1) + 1 = 3 + 2 = \binom{3}{2} + 2$
- $T(2,3) = T(1,3) + T(2,2) + 1 = 3 + (3+2) + 1 = 6 + 3 = \binom{4}{2} + 3$
- $T(2,4) = T(1,4) + T(2,3) + 1 = 4 + (6+3) + 1 = 10 + 4 = \binom{5}{2} + 4$

  $\vdots$

  $\vdots$
- $T(2,m) = T(1,m) + T(2,m-1) + 1 = \binom{m+1}{2} + m$

- $T(3,1) = T(2,1) + T(3,0) + 1 = 1 + 2 = 3 = \binom{3}{3} + 2 = \binom{3}{3} + T(2,1) = \binom{3}{3} + \binom{2}{2} + 1$
- $T(3,2) = T(2,2) + T(3,1) + 1 = 4 + 5 = 9 = \binom{4}{3} + 5 = \binom{4}{3} + T(2,2) = \binom{4}{3} + \binom{3}{2} + 2$
- $T(3,3) = T(2,3) + T(3,2) + 1 = 10 + 9 = 19 = \binom{5}{3} + 9 = \binom{5}{3} + T(2,3) = \binom{5}{3} + \binom{4}{2} + 3$
- $T(3,4) = T(2,4) + T(3,3) + 1 = 20 + 14 = 34 = \binom{6}{3} + 14 = \binom{6}{3} + T(2,4) = \binom{6}{3} + \binom{5}{2} + 4$

  $\vdots$

  $\vdots$
- $T(3,m) = T(2,m) + T(3,m-1) + 1 = \binom{m+2}{3} + \binom{m+1}{2} + m$

Since we have 2 variable inputs here - n, m for $G(n,m)$, it would be very helpful for us to draw a 2D matrix for proper analysis on test cases. In this matrix, the column numbers will start from 1 to m and row numbers will start from 1 to n.

$T(n,m)$	1	2	3	4	-	-	-	-	-	-	-	m − 1	m
1	1	2	3	4	-	-	-	-	-	-	-	m − 1	m
2	$1+1$	$3+2$	$6+3$	$10+4$	-	-	-	-	-	-	-	-	$\binom{m+1}{2} + m$
3	$1+2$	$4+5$	$10+9$	$20+14$	-	-	-	-	-	-	-	-	$\binom{m+2}{3} + \binom{m+1}{2} + m$
4	$1+3$												$\binom{m+3}{4} + \binom{m+2}{3} + \binom{m+1}{2} + m$
-													
-													
-													
$n-1$	$1+n-2$												$T(n,m) = \sum_{k=1}^{n-1} \binom{m+k-1}{k}$
n	$1+n-1$												$T(n,m) = \sum_{k=1}^{n} \binom{m+k-1}{k}$

Hence by seeing the above trend and in the below box, we can write:

$$T(n,m) = \binom{m+n-1}{n} + \binom{m+n-2}{n-1} + \binom{m+n-3}{n-2} + \ldots + \binom{m}{1}$$

We can write:

$$T(n,m) = \binom{m}{0} + \binom{m}{1} + \binom{m+1}{2} + \binom{m+2}{3} + \ldots + \binom{m+n-1}{n} - \binom{m}{0}$$

We know this:

$$\binom{n}{r} + \binom{n}{r+1} = \binom{n+1}{r+1}$$

Proof:

- $\binom{n}{r} = \frac{n!}{(n-r)!(r!)}$ and $\binom{n}{r+1} = \frac{n!}{(n-r-1)!(r+1)!}$
- $\binom{n}{r} + \binom{n}{r+1} = \frac{n!}{(n-r)!(r!)}\left(\frac{1}{n-r} + \frac{1}{r+1}\right) = \frac{n!}{(n-r)!(r!)} \cdot \frac{n+1}{(n-r)(r+1)} = \frac{(n+1)!}{(n-r)!(r+1)!} = \binom{n+1}{r+1}$

So, we can simplify:

- $T(n,m) = \binom{m+1}{1} + \binom{m+1}{2} + \binom{m+2}{3} + \ldots + \binom{m+n-1}{n-1} - \binom{m}{0}$
- $T(n,m) = \binom{m+2}{2} + \binom{m+2}{3} + \cdots + \binom{m+n-1}{n} - \binom{m}{0}$
- $T(n,m) = \binom{m+3}{3} + \cdots + \binom{m+n-1}{n} - \binom{m}{0}$

$\vdots$

$\vdots$

- $T(n,m) = \binom{m+n-1}{n-1} + \binom{m+n-1}{n} - \binom{m}{0}$
- $T(n,m) = \binom{m+n}{n} - \binom{m}{0}$

Therefore, the final expression for $T(n,m)$ is:

$$\boxed{T(n,m) = \binom{m+n}{n} - 1}$$

Now let's check the mathematical correctness of the above expression or solution :

[Proof by Mathematical Induction using single variable]

So here our proposition is : $\boxed{T(n,m) = \binom{m+n}{n} - 1 \text{ such that } n,m \text{ are positive integers}}$

To prove it by mathematical induction using single variable, we will do the following :

Basis step:

- $T(1,m)$ is true. Since by the recurrence relation we can see that $T(1,m) = 1 + T(0,m) + T(1,m-1) = 1 + T(1,m-1)$. And since this is true for all positive integer values of m, we can solve this recurrence relation like this :
  - $T(1,m) = 1 + T(1,m-1)$
  - We can write, $T(1,m-1) = 1 + T(n,m-2)$
  - Hence, we get : $T(1,m) = 1+1+1+1+1+...+T(1,1) = m$

  And by the above formula, $T(1,m) = \binom{m+1}{1} - 1 = m+1-1 = m$.
- $T(n,1)$ is true. Since by the recurrence relation we can see that $T(n,1) = 1 + T(n-1,1) + T(n,0) = 1 + T(n-1,1)$. And since this is true for all positive integer values of n, we can solve this recurrence relation like this :
  - $T(n,1) = 1 + T(n-1,1)$
  - We can write, $T(n-1,1) = 1 + T(n-2,1)$
  - Hence, we get : $T(n,1) = 1+1+1+1+1+...+T(1,1) = n$

  And by the above formula, $T(n,1) = \binom{n+1}{1} - 1 = n+1-1 = n$.

Hence, the basis step is proved.

Inductive step:

Assume that the recurrence holds for all $T(i,j)$ where $i+j<k$, i.e. $T(i,j) = \binom{i+j}{j} - 1$ for all $i+j<k$.

Now, we want to prove that $T(k-n,n) = \binom{k}{n} - 1$ for all $n<k$ i.e. when $i+j=k$.

Using the recurrence relation $T(n,m) = T(n,m-1) + T(n-1,m) + 1$, we can express $T(k-n,n)$ as follows:

- $T(k-n,n) = T(k-n,n-1) + T(k-n-1,n) + 1$
- Since $(k-n)+(n-1)$ and $(k-n-1)+(n)$ and $k-n-1$ are less than k, which means we can apply our induction hypothesis :
  - $T(k-n,n) = \binom{k-n+n-1}{n-1} - 1 + \binom{k-n-1+n}{n} - 1 + 1$
  - $T(k-n,n) = \binom{k-1}{n-1} + \binom{k-1}{n} - 1$
- Using the identity, $\binom{n}{r} + \binom{n}{r+1} = \binom{n+1}{r+1}$, we can rewrite the expression:

  $T(k-n,n) = \binom{k-1}{n-1} + \binom{k-1}{n} - 1 = \binom{k}{n} - 1$
- Thus, by induction, the given recurrence relation $T(n,m) = T(n,m-1) + T(n-1,m) + 1$ holds for all $n+m$.

[Proof by Mathematical Induction using two variables]

So here our proposition is : $\boxed{T(n,m) = \binom{m+n}{n} - 1 \text{ such that } n, m \text{ are positive integers}}$

Basis step:
$T(1,1)$ is true. Since by the recurrence relation we can see that $T(1,1) = 1 + T(0,1) + T(1,0) = 1$

And by the above formula, $T(1,1) = \binom{2}{1} - 1 = 1$.

Hence, the basis step is proved.

Inductive step:
Assume that $T(1,1)$, $T(1,2)$, $T(1,3)$, $T(1,4)$, …, $T(1,m)$, $T(2,1)$, $T(2,2)$, …, $T(2,m)$, …, $T(n,1)$, $T(n,2)$, …, $T(n,m-1)$ are true for any arbitrary integers $n > 1$ and $m > 1$. We can write because we are working with 2 integer variables - n and m. And we need to prove that $T(n,m)$ holds true.

- Since we have assumed that $T(n-1,m)$ and $T(n,m-1)$ to be true. Hence,
  - $T(n-1,m) = \binom{m+n-1}{n-1} - 1$
  - $T(n,m-1) = \binom{m+n-1}{n} - 1$
- By the recurrence relation which we have written at the start of the solution, i.e.,
  - $T(n,m) = 1 + T(n-1,m) + T(n,m-1)$

  We can write $T(m,n) = 1 + \binom{m+n-1}{n-1} - 1 + \binom{m+n-1}{n} - 1$.
- Using $\binom{n}{r} + \binom{n}{r+1} = \binom{n+1}{r+1}$,
  - $T(m,n) = \binom{m+n-1}{n-1} + \binom{m+n-1}{n} - 1 = \binom{m+n}{n} - 1$.
- Hence, we have proved that $T(m,n)$ also satisfies the equation.

2. (*Proof techniques review*) Let us quickly review a few proof techniques you must have studied in your introductory Mathematics course.

- Direct proof: *Used for showing statements of the form p implies q. We assume that p is true and use axioms, definitions, and previously proven theorems, together with rules of inference, to show that q must also be true.*
- Proof by contraposition: *Used for proving statements of the form p implies q. We take $\neg q$ as a premise, and using axioms, definitions, and previously proven theorems, together with rules of inference, we show that $\neg p$ must follow.*
- Proof by contradiction: *Suppose we want to prove that a statement p is true and suppose we can find a contradiction q such that $\neg p$ implies q. Since q is false, but $\neg p$ implies q, we can conclude that $\neg p$ is false, which means that p is true. The contradiction q is usually of the form $r \wedge \neg r$ for some proposition r.*
- Counterexample: *Suppose we want to show that the statement for all x, $P(x)$ is false. Then we only need to find a counterexample: an example x for which $P(x)$ is false.*
- Mathematical induction: Used for proving statements of the form $\forall n, P(n)$. This has already been discussed in class.

Solve the proof problems that follow:

(a) (3 points) Give a direct proof of the statement: "If n is odd, then n^2 is odd".

[**Solution**]

Here let's take the statement p to be: n is odd. And the statement q to be: n^2 is odd. We need to assume the statement p to be true in Direct Proof Method. Therefore, we can write n as $2k + 1$ where k can be any integer (by the definition of an odd integer).

Assuming this is true, let's find out $n^2 = (2k+1)^2 = 4k^2+4k+1 = 2(2k^2+2k)+1 = 2m+1$. Here $m = 2k^2 + 2k$ where m is also an integer as k is also an integer according to the above explanation of an odd integer ($n = 2k + 1$). So we arrived at $n^2 = 2m + 1$, where m is an integer. Hence we can say, by the definition of an odd integer, that n^2 is an odd integer. Hence proved.

(b) (3 points) Prove by contraposition that "if n^2 is odd, then n is odd".

[**Solution**]

Here let's take the statement p to be: n^2 is odd. And the statement q to be: n is odd. We need to assume the statement $\neg q$ to be premise i.e we need to assume it true for proving by Contraposition. Now $\neg q$ statement will be : n is even (as if n is not odd integer then it has to be even integer by definition of integers). Therefore, we can write n as $2k$ where k can be any integer (by the definition of an even integer).

Now let's find out $n^2 = (2k)^2 = 4k^2 = 2(2k^2) = 2m$ Here $m = 2k^2$ where m is also an integer as k is also an integer according to the above explanation of an even integer ($n = 2k$). So we arrived at $n^2 = 2m$, where m is an integer. Hence we can say, by the definition of an even integer, that n^2 is an even integer. Hence proved.

(c) (3 points) Give proof by contradiction of the statement: "at least four of any 22 days must fall on the same day of the week."

[**Solution**]

Let p be the proposition - "at least four of any 22 days must fall on the same day of the week." Suppose that **¬p** is true.

$$\boxed{\textbf{¬p}\text{: ``at most three of any 22 days must fall on the same day of the week.''}}$$

Since there are 7 days in a week, this implies that at most 21 days could have been chosen (because at most 3 of the chosen days could fall on that day for each of the days of the week). But since we have 22 days under consideration, this means there must be atleast one day of the week which has 4 days falling on it (because of only 7 days in a week).

Hence **¬p** implies that there must be atleast one day of the week which has 4 days falling on it.

Lets take **q** : there must be atleast one day of the week which has 4 days falling on it.

q is false as we have taken atmost 3 of the chosen days could fall on the same day of the week. Hence we conclude that **¬p** is false. This means p is true. Hence we have proved that "at least four of any 22 days must fall on the same day of the week."

(d) (3 points) Use a counterexample to show that the statement "Every positive integer is the sum of squares of two integers" is false.

[**Solution**]

To show that the statement "Every positive integer is the sum of squares of two integers" is false, we need to find a positive integer that cannot be expressed as the sum of squares of two integers.
Let's take the positive integer 6 as an example. We need to check if there exist two integers, let's call them a and b, such that:

$$6 = a^2 + b^2.$$

Now, let's try all possible combinations of squares of integers that are less than or equal to 6:

$$\begin{aligned} 1^2 + 1^2 &= 2 \quad \text{(not equal to 6)} \\ 1^2 + 2^2 &= 5 \quad \text{(not equal to 6)} \\ 2^2 + 1^2 &= 5 \quad \text{(not equal to 6)} \\ 2^2 + 2^2 &= 8 \quad \text{(not equal to 6)} \end{aligned}$$

For all the other combinations of a and b, the value of $a^2 + b^2$ will always be greater than 6. Since none of these combinations result in 6, we can conclude that 6 cannot be expressed as the sum of squares of two integers. Therefore, the statement "Every positive integer is the sum of squares of two integers" is false, and 6 serves as a counterexample.

(e) (3 points) Show using mathematical induction that for all $n \geq 0$, $1+\frac{1}{2^1}+\frac{1}{2^2}+\frac{1}{2^3}+...+\frac{1}{2^n} = \frac{1-(\frac{1}{2})^{n+1}}{1-\frac{1}{2}}$.

[**Solution**]

[Proof by Mathematical Induction]

So here our proposition is :

$\boxed{T(n) = 1 + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \ldots + \frac{1}{2^n} = \frac{1-\left(\frac{1}{2}\right)^{n+1}}{1-\frac{1}{2}}}$, for all $n > 0$

Basis step:
The given equation is: $T(1)$ is true. Since by the above relation we can see that:

$$\text{LHS} = 1 + \frac{1}{2^1} = \frac{3}{2} \quad \text{and} \quad \text{RHS} = \frac{1-\left(\frac{1}{2}\right)^2}{1-\frac{1}{2}} = \frac{\frac{3}{4}}{\frac{1}{2}} = \frac{3}{2}$$

Hence, the basis step is proved.

Inductive step:
Assume that $T(1), T(2), \ldots, T(k)$ are true for any arbitrary integer $k > 1$.

- We can write $T(k+1) = 1 + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \ldots + \frac{1}{2^k} + \frac{1}{2^{k+1}}$.
- Since we have assumed $T(k)$ to be true, therefore
  $T(k) = 1 + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \ldots + \frac{1}{2^k} = \frac{1-(\frac{1}{2})^{k+1}}{1-\frac{1}{2}}$.
- Putting above expression in $T(k+1)$, we get $T(k+1) = \frac{1-(\frac{1}{2})^{k+1}}{1-\frac{1}{2}} + \frac{1}{2^{k+1}}$
  Simplifying it, we get $T(k+1) = 2 - (\frac{1}{2^k}) + \frac{1}{2^{k+1}} = 2 - \frac{1}{2^{k+1}} = \frac{1-(\frac{1}{2})^{k+2}}{1-\frac{1}{2}}$
- So, $T(k+1) = \frac{1-(\frac{1}{2})^{k+2}}{1-\frac{1}{2}}$. Hence we have proved that $T(k+1)$ also is true because RHS in case of $T(k+1)$ is also $\frac{1-(\frac{1}{2})^{k+2}}{1-\frac{1}{2}}$

3. (*Working with the definition of big-O*)

Big-O notation requires practice to become comfortable. There is a bit of mathematical jugglery since the definition involves a quantified statement. However, the mathematics involved should not be new to you. All that is required is to work with the quantified statement. Let us consider an example. Suppose there is an exponential-time algorithm with a running time expression 2^n. Would it be correct to say that the running time is $O(3^n)$? Yes, this would be correct. Let us work with the definition to verify this. To show that 2^n is $O(3^n)$, all we need to do is to give constants $c > 0$ and $n_0 > 0$ such that $\forall n \geq n_0, 2^n \leq c \cdot 3^n$. It is easy to check that this holds for constants $c = 1$ and $n_0 = 1$.

Let us consider the reverse case. That is, suppose the running time of an algorithm is 3^n. Would it be correct to say that the running time is $O(2^n)$? No, this would not be correct. The way to argue that 3^n is **not** $O(2^n)$ is to show the negation of the quantified statement for this example. That is, we need to show that for **any** constants $c > 0, n_0 > 0$, there exists $n \geq n_0$ such that $3^n > c \cdot 2^n$. Note that $3^n > c \cdot 2^n \Leftrightarrow (3/2)^n > c \Leftrightarrow n > \log_{3/2} c$. So, for any constant c, $3^n > c \cdot 2^n$ when $n > \log_{3/2} c$. So, for any constants $c > 0$ and $n_0 > 0$, let us try $n' = \max(\lceil \log_{3/2} c + 1 \rceil, n_0)$. We find that $3^{n'} > c \cdot 2^{n'}$ and furthermore $n' \geq n_0$. From this, we conclude that 3^n is not $O(2^n)$.

Prove or disprove the following statements:

(a) (3 points) $2^{\sqrt{\log n}}$ is $O(n)$.

[**Solution**]

To show that $2^{\sqrt{\log n}}$ is $O(n)$, all we need to do is to give constants $c > 0$ and $n_0 > 0$ such that $\forall n \geq n_0$, $2^{\sqrt{\log n}} \leq c \cdot n$.

- Taking logarithm (with base 2 always in this assignment) on both sides of $2^{\sqrt{\log n}} \leq c \cdot n$ we get $\sqrt{\log n} \leq \log c + \log n$, as both sides of inequality are positive entities and logarithm is increasing function.
- Since we have to find $c > 0$, let's take $c = 1$, so now we have the inequality as $\sqrt{\log n} \leq \log n$
- Because we perform asymptotic analysis on large values of n, we can square terms on both sides of inequality : $\log n \leq (\log n)^2$
- And above inequality is true for all $n > 2$ as $log 2 = 1$ and for $n > 2$, $\log n$ value increases and will always be > 1
- Hence we got values of $c > 0$ and $n_0 > 0$ i.e $c = 1$ and $n_0 = 2$ and the inequality $2^{\sqrt{\log n}} \leq c \cdot n$ hold $\forall n \geq n_0$ and for $c = 1$

Hence we have proved : $2^{\sqrt{\log n}}$ is $O(n)$

(b) (7 points) $(9n2^n + 3^n)$ is $\Omega(n3^n)$.

[**Solution**]

Let's try to disprove this statement and see whether we are able to reach at solution or not.

- To show that $(9n2^n + 3^n)$ is not $\Omega(n3^n)$, we need to show that for any constants $c > 0$ (real constant) and $n_0 > 0$ (integer constant), there exists an $n \geq n_0$ such that $(9n \cdot 2^n + 3^n) < c \cdot n \cdot 3^n$, we need to find a suitable value of n that satisfies this inequality for any given c and n_0.
- Let's proceed with the proof:

  Consider the expression $(9n \cdot 2^n + 3^n)$ and the term $c \cdot n \cdot 3^n$.

  We want to find an $n \geq n_0$ such that $(9n \cdot 2^n + 3^n) < c \cdot n \cdot 3^n$ for any $c > 0$ and n_0.
- Now, we aim to find a suitable n_0 such that for all $n \geq n_0$:

  $(9n \cdot 2^n + 3^n) < c \cdot n \cdot 3^n$

  Let's try to solve this inequality for n:

  $\frac{(9n \cdot 2^n + 3^n)}{3^n} < c \cdot n = 9n \cdot \frac{2^n}{3^n} + 1 < c \cdot n = \frac{9n}{(3/2)^n} + 1 < c \cdot n$

  Since $\frac{9n}{(3/2)^n}$ approaches 0 as n approaches infinity, we can ignore this term for sufficiently large n (when $n \geq n_0$).

  So, we have:

  $1 < c \cdot n$

  Divide both sides by n:

  $\frac{1}{c} < n$

  Since $c > 0$, we can set $n_0 = \frac{1}{c}$. For all $n \geq n_0$ $(n \geq \frac{1}{c})$, the inequality holds:

  $(9n \cdot 2^n + 3^n) \leq c \cdot n \cdot 3^n$

  This proves that for all $c > 0$ and $n_0 = \frac{1}{c}$, there exists an $n \geq n_0$ such that $(9n \cdot 2^n + 3^n) < c \cdot n \cdot 3^n$.

This means that we have proved that $(9n2^n + 3^n)$ is not $\Omega(n3^n)$. Hence we have disproved the original statement.

4. (*More practice with big-O definition*) Prove or disprove the following statements:

   (a) (5 points) If $d(n)$ is $O(f(n))$ and $f(n)$ is $O(g(n))$, then $d(n)$ is $O(g(n))$.

   [**Solution**]

   - Since $d(n) \in O(f(n))$, there exist positive constants $c_1 > 0$ and $n_1 > 0$ such that $d(n) \le c_1 \cdot f(n)$ for all $n \ge n_1$.
   - Also, since $f(n) \in O(g(n))$, there exist positive constants $c_2 > 0$ and $n_2 > 0$ such that $f(n) \le c_2 \cdot g(n)$ for all $n \ge n_2$.
   - Let's choose $c_3 = c_1 \cdot c_2$ and $n_3 = \max(n_1, n_2)$. Then for all $n \ge n_3$, we have: $d(n) \le c_1 \cdot f(n) \le c_1 \cdot (c_2 \cdot g(n)) = c_3 \cdot g(n)$.

     Thus, we have shown that $d(n) \in O(g(n))$ with constants c_3 and n_3, which completes the proof.

   (b) (5 points) $\max\{f(n), g(n)\}$ is $O(f(n) + g(n))$.

   [**Solution**]

   - Let $h(n) = \max\{f(n), g(n)\}$.
   - We know that $h(n) \le f(n) + g(n)$, as the maximum of two numbers is always less than or equal to their sum. And we can safely assume $f(n)$ and $g(n)$ as positive functions because with that purpose of analysing positive functions, we derived the definition of $O(n)$
   - Now, let's find positive constants c and n_0 such that $h(n) \le c \cdot (f(n) + g(n))$ for all $n \ge n_0$.
   - Since $h(n) \le f(n) + g(n)$, we have: $h(n) \le 1 \cdot (f(n) + g(n))$ for all n.

     So, we can choose $c = 1$ and any n_0 to satisfy the condition.

     Therefore, $\max\{f(n), g(n)\}$ is $O(f(n) + g(n))$.

   (c) (5 points) If $a(n)$ is $O(f(n))$ and $b(n)$ is $O(g(n))$, then $a(n) + b(n)$ is $O(f(n) + g(n))$.

   [**Solution**]

   - Since $a(n)$ is $O(f(n))$, there exist positive constants $c_1 > 0$ and $n_1 > 0$ such that for all $n \ge n_1$:
     $$a(n) \le c_1 \cdot f(n).$$
   - Similarly, since $b(n)$ is $O(g(n))$, there exist positive constants $c_2 > 0$ and $n_2 > 0$ such that for all $n \ge n_2$:
     $$b(n) \le c_2 \cdot g(n).$$

- Let $n_0 = \max(n_1, n_2)$ and $c = c_1 + c_2$. For all $n \geq n_0$, we have:

$$a(n) + b(n) \leq c_1 \cdot f(n) + c_2 \cdot g(n).$$

- Since $c_1 \cdot f(n) \leq c \cdot f(n)$ and $c_2 \cdot g(n) \leq c \cdot g(n)$, we can write:

$$a(n) + b(n) \leq c \cdot f(n) + c \cdot g(n) = c \cdot (f(n) + g(n)).$$

  Therefore, $a(n) + b(n)$ is $O(f(n) + g(n))$ with $c = c_1 + c_2$ and n_0 as the positive constants.

(d) (5 points) If $f(n)$ is $O(g(n))$, then $2^{f(n)}$ is $O(2^{g(n)})$.

[**Solution**]

- Since $f(n)$ is $O(g(n))$, there exist positive constants $c_1 > 0$ and $n_1 > 0$ such that for all $n \geq n_1$:

$$f(n) \leq c_1 \cdot g(n).$$

- Let $n_0 = n_1$ and $c = 2^{c_1}$. For all $n \geq n_0$, we have:

$$2^{f(n)} \leq 2^{c_1 \cdot g(n)}.$$

- So, we got $2^{f(n)} \leq 2^{c_1} \cdot 2^{g(n)}$. Therefore, $2^{f(n)}$ is $O(2^{g(n)})$ with $c = 2^{c_1}$ and $n_0 = n_1$ as the positive constants.

(e) (5 points) If $f(n)$ is $O(g(n))$, then $f(2n)$ is $O(g(2n))$.

[**Solution**]

- Since $f(n)$ is $O(g(n))$, there exist positive constants $c_1 > 0$ and $n_1 > 0$ such that for all $n \geq n_1$:

$$f(n) \leq c_1 \cdot g(n).$$

- Let $h(n) = f(2n)$. Here take the same conditions i.e for all $n \geq n_1$. So, $2n \geq 2n_1 > n_1$. Hence for same constant $c_1 > 0$, we can see easily that $f(2n) \leq c_1 \cdot g(2n)$ since $2n > n1$
- So, $f(2n)$ is $O(g(2n))$.

5. (15 points) (*Arguing correctness using induction*) Argue using induction that the following algorithm correctly computes the value of a^n when given positive integers a and n as input.

```
Exponentiate(a, n)
    · if ($n = 0$) return(1)
    · $t$ = Exponentiate($a$, $\lfloor n/2 \rfloor$)
    · if ($n$ is even) return($t \cdot t$)
    · else return($t \cdot t \cdot a$)
```

[**Solution**]

Now let's check the mathematical correctness of the above expression or solution :

[Proof by Mathematical Induction]

So here our proposition is : $\boxed{\texttt{Exponentiate(a, n)} = a^n \text{ where a and n are positive integers}}$

Basis step:

`Exponentiate(a, 1)` is true. Since by the recurrence relation we can see that :

- t = `Exponentiate(a, 0)` = 1
- Since 1 is odd integer, so `Exponentiate(a, 1)` = 1.1.a = a

And by the above formula, `Exponentiate(a, n)` $= a^n = a^1 = a$

Hence, the basis step is proved.

Inductive step:

Assume that `Exponentiate(a, 1)`, `Exponentiate(a, 2)`, `Exponentiate(a, 3)`, . . ., `Exponentiate(a, k)` are true for any arbitrary integers $k > 1$. And we need to prove that `Exponentiate(a, k+1)` holds true.

Let's take `Exponentiate(a, k+1)` and derive the expression for it. Here there will be 2 cases for (k+1).

- (k+1) is odd integer :
  - By the recurrence relation given in the question, we can write :
    $t =$ `Exponentiate(a,` $\lfloor \frac{k+1}{2} \rfloor$`)`
  - Now, $\lfloor \frac{k+1}{2} \rfloor = \lfloor \frac{k}{2} + \frac{1}{2} \rfloor = \frac{k}{2}$ as $\frac{k}{2}$ will be integer and $\frac{1}{2}$ will not make any change to value of floor function as value inside floor function will be of form $(m + 0.5)$ where m is an integer.
  - t = `Exponentiate(a,` $\frac{k}{2}$`)` $= a^{\frac{k}{2}}$ as we have assumed `Exponentiate(a, 1)`, `Exponentiate(a, 2)`, `Exponentiate(a, 3)`, . . ., `Exponentiate(a, k)` as true.
  - So, we now got the value of t = `Exponentiate(a,` $\lfloor \frac{k+1}{2} \rfloor$`)` $= a^{\frac{k}{2}}$. Further by seeing recurrence relation, we can see that since (k+1) is odd, so the value of the function `Exponentiate(a, k+1)` = t.t.a = $a^{\frac{k}{2}}.a^{\frac{k}{2}}$.a $= a^{k+1}$. Hence we have proved that `Exponentiate(a, k+1)` $= a^{k+1}$ when (k+1) is odd.

- (k+1) is even integer :
  - By the recurrence relation given in the question, we can write :
    $t =$ `Exponentiate(a,` $\lfloor\frac{k+1}{2}\rfloor$`)` $=$ `Exponentiate(a,` $\frac{k+1}{2}$`)` as (k+1) is an even integer
  - t $=$ `Exponentiate(a,` $\frac{k+1}{2}$`)` $= a^{\frac{k+1}{2}}$ as we have assumed `Exponentiate(a, 1)`, `Exponentiate(a, 2)`, `Exponentiate(a, 3)`, $\ldots$, `Exponentiate(a, k)` for $k > 1$ as true.
  - So, we now got the value of $t =$ `Exponentiate(a,` $\lfloor\frac{k+1}{2}\rfloor$`)` $= a^{\frac{k+1}{2}}$. Further by seeing recurrence relation, we can see that since (k+1) is even, so the value of the function `Exponentiate(a, k+1)` $=$ t.t $= a^{\frac{k+1}{2}}.a^{\frac{k+1}{2}} = a^{k+1}$. Hence we have proved that `Exponentiate(a, k+1)` $= a^{k+1}$ when (k+1) is even.

6. (*Analysing running time*) We start with a brief discussion.

As discussed in class, the big-O notation allows us to ignore the hairy details we may need to keep track of otherwise. Let us consider the example of the Insertion Sort algorithm below for this discussion. `InsertionSort(A, n)`

```
· for $i = 1$ to $n$
  · $j = i - 1$
  · while($j > 0$ and $A[j] > A[i]$) $j$- -
  · for $k = j$+1 to $i$-1
    · Swap $A[i]$ and $A[k]$
```

The first line of the algorithm is a 'for' statement. How many basic operations does this 'for' statement contribute? Getting a precise count will require some deeper analysis of its implementation mechanism. It should involve a comparison operation in every iteration since one needs to check if the variable i has exceeded n. It should include arithmetic operation since the variable i needs to be incremented by 1 at the end of each iteration. Are these all the operations? Not really. The increment operation involves loading the variable and storing it after the increment. Keeping track of all these hairy details may be too cumbersome. A quick way to account for all possible basic operations is to say that the contribution to the number of operations coming from every iteration of the outer 'for' statement is a constant. So, the number of operations contributed by the outer 'for' statement is $an + b$ for some constants a, b. This is the level of granularity at which we shall do the counting to keep things simple.

Answer the questions that follow:

(a) (2 points) What is the worst-case number of operations contributed by the while-loop in the i^{th} iteration of the outer for-loop as a function of i? (*You can give an expression in terms of symbols for constants as we did for the outer 'for' statement in the discussion. This will be sufficient since the big-O will allow us to ignore the specific constants.*)

[**Solution**]

In the ith iteration of the outer 'for'-loop, for calculating worst-case number of operations contributed by the 'while'-loop present inside, let's see this part of the code:

while($j > 0$ and $A[j] > A[i]$) j- -

Here from above, we can see that since $j = i - 1$; for the worst case, the 'while' loop condition will get evaluated each time till $j > 0$, i.e., a total of $(i - 1)$ times as we can take $A[j]$ to be always greater than $A[i]$ for decreasing j from $i - 1$ to 1 as each time the loop condition is satisfied, j is also decremented by one.
Let's take time to perform :

- this decrement operation takes b (constant).
- Condition checking in the 'while' loop takes a (constant).

Hence, the total worst-case number of operations contributed by the 'while'-loop in the ith iteration of the outer 'for'-loop as a function of i will be equal to $(a + b)(i - 1) + d$ (some other constant) which is of the form $\boxed{(\alpha i + \beta)}$.

(b) (2 points) Similarly, what is the worst-case number of operations contributed by the inner for-loop in the i^{th} iteration of the outer for-loop as a function of i? Again, express using symbolic constants.

[**Solution**]

Similarly, as discussed in the above part, we can perform similar analysis on this for loop :

```
· for $k$ = $j$+1 to $i$-1
  · Swap $A[i]$ and $A[k]$
```

In the worst case, at the end of the previous while loop, the value of $j = 0$. Hence, we can see that the 'for' loop will run for at most $(i - 1)$ times. And whenever we go inside the for loop, we need to perform a swap operation, which will take constant time operations as the swap operation will get performed in 3 steps, and we can assume constant operations for these 3 steps: i.e.,

$$\begin{aligned} t &= A[i] \\ A[i] &= A[k] \\ A[k] &= t \end{aligned}$$

The total worst-case number of operations contributed by the inner 'for' - loop in the ith iteration of the outer for-loop as a function of i will be equal to $e(i - 1) + f$ (some other constant). Which is of the form $\boxed{\alpha_1 \text{ i} + \beta_1}$.

(c) (2 points) Use expressions of the previous two parts to give the worst-case number of operations executed by the algorithm. Use big-O notation. Give a brief explanation.
[**Solution**]

The asymptotic - worst - case number of operations contributed by the 'while'-loop is $O(i)$.

The asymptotic - worst - case number of operations by the inner 'for' loop is $O(i)$.

So, for each i in the outer 'for' loop, the asymptotic - worst - case number of operations performed will be $O(1)$ (for $j = i - 1$ operation) + $O(i)$ + $O(i) = O(i)$.
Considering the outer 'for' loop, the total number of asymptotic - worst - case operations in performed can be expressed as:

$$\begin{aligned} \text{Total operations} &= O(1) + O(2) + O(3) + \ldots + O(n) \\ &= O\left(\sum_{i=1}^{n} i\right) \\ &= O\left(\frac{n(n+1)}{2}\right) \\ &= O(n^2) \end{aligned}$$

Therefore, the worst-case time complexity of this algorithm is $\boxed{O(n^2)}$.

(d) (4 points) Show that your running time analysis in the previous part is tight.
[**Solution**]

For showing running time analysis that we did in the previous part is tight, we need one example for which the number of operations performed is $O(n^2)$ in asymptotic analysis. That example is:

- Let's take the array of n-elements which is sorted in decreasing order and apply the insertion sort algorithm on it.
  - When the array is in decreasing order, inside the outer 'for' loop, the 'while' condition will always get evaluated, so the number of operations performed by the 'while' loop will also be $O(i)$ for the ith iteration of the outer 'for' loop.
  - The final value of j after the execution of the 'while' loop will be 0. Hence, the number of operations performed by the inner 'for' loop will also be $O(i)$ for the ith iteration of the outer 'for' loop.

Hence, on analysing just like we did in the previous subpart, we can see that the number of operations performed is $O(n^2)$ in asymptotic analysis in case when we are having an array of n elements in decreasing order.