Question 1.

For $k \in \mathcal{K}, n \in \mathcal{N}, m = 1, \ldots, M$, we define the binary variables $x_{k,m,n} \in \{0,1\}$ such that $x_{k,m,n} = 1$ if and only if $p_{k,m,n} \leq p_{k,n} < p_{k,m+1,n}$, with $p_{k,M+1,n}$ interpreted as $+\infty$. Thus, the constraint by the total power budget is

$$\sum_{\substack{k \in \mathcal{K} \\ n \in \mathcal{N} \\ m=1,\ldots,M}} p_{k,m,n} x_{k,m,n} \leq p$$

the constraint that each channel serves one and only one user is

$$\sum_{\substack{k \in \mathcal{K} \\ m=1,\ldots,M}} x_{k,m,n} = 1, \, \forall n \in \mathcal{N}$$

and the target function is

$$U := \sum_{\substack{k \in \mathcal{K} \\ n \in \mathcal{N} \\ m=1,\ldots,M}} r_{k,m,n} x_{k,m,n}$$

In all, we have the ILP below

$$x_{k,m,n} \in \{0,1\} \tag{1}$$

$$\sum_{k,m,n} p_{k,m,n} x_{k,m,n} \leq p \tag{2}$$

$$\sum_{k,m} x_{k,m,n} = 1, \, \forall n \in \mathcal{N} \tag{3}$$

with target function

$$U := \sum_{k,m,n} r_{k,m,n} x_{k,m,n}$$

the corresponding LP is obtained by replacing (1) with $x_{k,m,n} \in [0,1]$

Question 2.

Proof of **Lemma 1**

Suppose $p_{k,m,n} \leq p_{k',m',n}$ and $r_{k,m,n} \geq r_{k',m',n}$. Given an optimal solution to the ILP, if $x_{k',m',n} \neq 0$, then $x_{k',m',n} = 1$ by (1). If we replace $x_{k,m,n}$ by 1 and $x_{k',m',n}$ by 0, (1) and (3) are clearly satisfied, (2) is also satisfied since $p_{k,m,n} \leq p_{k',m',n}$, and U will not decrease since $r_{k,m,n} \geq r_{k',m',n}$, so we get an optimal where $x_{k',m',n} = 0$

Question 3.

REMOVE-IP-DOMINATED(n)

1 sort the pairs $(p_{k,m,n}, r_{k,m,n})$ in increasing order of $p_{k,m,n}$ into an array A. If several pairs have the same p, leave only the one with the greatest r

2 cm = A[0].r

3 **for** i = 0 to A.length-1

4   **if** A[i].r $\geq$ cm

5   cm = A[i].r

6   **else** remove A[i].p and A[i].r from the original data

sorting $p_{k,m,m}$ takes time $O(KM\log(KM))$, the loop from line 3 takes time $O(KM)$. We will run REMOVE-IP-DOMINATED for each $n \in \mathcal{N}$, so in all it takes time $O(NKM\log(KM))$ to remove IP-dominated terms.
N.B. Accounting for that $p_{k,m,n}$ is increasing according to m for k, n fixed, we may sort quicker in line 1 and achieve a complexity of $O(NKM\log(K)$.

Question 4.

Proof of **Lemma 2**

Suppose $p_{k,m,n} \leq p_{k',m',n} \leq p_{k'',m'',n}$ and

$$\frac{r_{k'',m'',n} - r_{k',m',n}}{p_{k'',m'',n} - p_{k',m',n}} \geq \frac{r_{k',m',n} - r_{k,m,n}}{p_{k',m',n} - p_{k,m,n}} \tag{4}$$

note that we have

$$p_{k',m',n} = p_{k,m,n}\frac{p_{k'',m'',n} - p_{k',m',n}}{p_{k'',m'',n} - p_{k,m,n}} + p_{k'',m'',n}\frac{p_{k',m',n} - p_{k,m,n}}{p_{k'',m'',n} - p_{k,m,n}} \tag{5}$$

and from (4) we can deduce that

$$r_{k',m',n} \leq r_{k,m,n}\frac{p_{k'',m'',n} - p_{k',m',n}}{p_{k'',m'',n} - p_{k,m,n}} + r_{k'',m'',n}\frac{p_{k',m',n} - p_{k,m,n}}{p_{k'',m'',n} - p_{k,m,n}} \tag{6}$$

Given an optimal solution to the LP, we can construct another solution with

$$\begin{aligned}
x'_{k,m,n} &= x_{k,m,n} + x_{k',m',n}\frac{p_{k'',m'',n} - p_{k',m',n}}{p_{k'',m'',n} - p_{k,m,n}} \\
x'_{k',m',n} &= 0 \\
x'_{k'',m'',n} &= x_{k'',m'',n} + x_{k',m',n}\frac{p_{k',m',n} - p_{k,m,n}}{p_{k'',m'',n} - p_{k,m,n}}
\end{aligned}$$

(3) is satisfied since $x'_{k,m,n} + x'_{k',m',n} + x'_{k'',m'',n} = x_{k,m,n} + x_{k',m',n} + x_{k'',m'',n}$, and so is (2) since $x'_{k,m,n}$, $x'_{k',m',n}$, $x'_{k'',m'',n} \geq 0$ and none of them can surpass 1 or else one of the $x'\cdot, \cdot, n$ would be negative.

(2) is satisfied owing to (5), and $U' \geq U$ owing to (6). Thus we get an optimal solution where $x_{k',m',n} = 0$

In the pseudo-code below, we consider the input as points in the plane with coordinates $(p_{k,m,n}, r_{k,m,n})$. For simplicity, for a stack S, we note S[0] the top element and S[1] the second top one; for tow points A, B in the plane, we note $L(A, B)$ the slope of the line formed between them; we say B is *dominated* by A and C if and only if $A.p \leq B.p \leq C.p$ and $L(A, B) \leq L(B, C)$, which is just another interpretation of (4)

REMOVE-LP-DOMINATED(n)

1 sort the pairs $(p_{k,m,n}, r_{k,m,n})$ in increasing order of $p_{k,m,n}$ into an array A. If several pairs have the same p, leave only the one with the greatest r

2 let S be a stack

3 PUSH(A[0], S)

4 PUSH(A[1], S)

5 **for** i = 2 to A.length-1

6   **while** L(A[i],S[0]) $\geq$ L(S[0],S[1])

7   POP(S)

8   PUSH(A[i], S)

9 **return** S

Proof of correctness We use the loop invariant that after each iteration of line 5, S contains exactly all the points that are not dominated by points from A[0] to A[i], the line segments form a convex curve.

The invariant trivially holds before line 5. Suppose it holds before the ith iteration. During the ith iteration, the points removed by line 7 are clearly dominated by A[i] and S[1]. After the **while** loop terminates, we can be sure that all points in A between S[0] and A[i] are dominated by these two points and thus cannot be added to S, nor can the points between A[0] and S[0] by the loop invariant.

Note that the ponits in S form a convex curve, so that by L(A[i],S[0]) $<$ L(S[0],S[1]) we have that after line 8 the points in S still form a convex curve and so no points in S are dominated by points from A[0] to A[i], and by the arguments above, these are exactly all the points that have this property.

Time complexity Line 1 takes time $O(KM \log(KM))$. Regarding the **for** loop from line 5 to 8, note that each point can only be pushed or popped only once, so in all the **for** loop takes time $O(KM)$. We run the algorithm for each $n \in \mathcal{N}$, so altogether it takes time $O(NKM \log(KM))$ to remove all LP-dominated terms.

Question 5.

TO DO After coding

Question 6.

Greedy Algorithm We consider establishing a greedy solution. As indicated in the question stem, for each n, we sort the $(p_{l,n}, r_{l,n})$ in the ascending order of $p_{l,n}$. We add one more convention $\forall n, p_{0,n} = 0, r_{0,n} = 0$. Initially, we allocate to each channel no user, which means $(p_{0,n}, r_{0,n})$. Thus the initial utility is 0.

To increase the total utility, every loop, we choose one channel to allocate more power. The criteria is the $e_{l,n} = \dfrac{r_{l+1,n} - r_{l,n}}{p_{l+1,n} - p_{l,n}}$, which shows the average payoff we can get by increase l to $l+1$. So each time we choose the channel with the largest $e_{l,n}$ and change its user from l to $l+1$. The loop ends, when all channels have been allocated with the users with the largest power or $p \leq p_{current}$

When $p < p_{current}$, the current allocation is not feasible. So we allocate to the last channel a linear combination of its last two allocation. We have

$$\sum_{k \neq n} p_{l_k,k} + p_{l-1,n} < p \tag{7}$$

$$\sum_{k \neq n} p_{l_k,k} + p_{l,n} = p_{current} > p \tag{8}$$

We search for a ϵ that

$$\sum_{k \neq n} p_{l_k,k} + \epsilon p_{l-1,n} + (1-\epsilon) p_{l,n} = p \tag{9}$$

Combining (8) and (9), we get ϵ

$$\epsilon = \frac{p_{current} - p}{p_{l,n} - p_{l-1,n}}$$

GREEDY-ALGORITHM-SOLUTION(n)

$p_{current} = 0$
for n = 1 to N
    $l[n] = 0$
while $p_{current} < p$
    find the $n \in 1, ...N$ with the largest $e_{l[n]n} = \dfrac{r_{l[n]+1,n} - r_{l[n],n}}{p_{l[n]+1,n} - p_{l[n],n}}$
    $p_{current} += p_{(l[n_m]+1),n_m} - p_{l[n_m],n_m}$
    l[n] += 1;

8 $\quad n_m = n$
9 **if** $p_{current} == p$
10 $\quad$ **for** n = 1 to N
11 $\quad\quad x_{l[n],n} = 1$
12 **else**
13 $\quad$ **for** n = 1 to N **not** n_m
14 $\quad\quad x_{l[n],n} = 1$
15 $\quad x_{l[n_m]-1,n_m} = \epsilon := \dfrac{p_{current} - p}{p_{l[n_m],n_m} - p_{l[n_m]-1,n_m}}$
16 $\quad x_{l[n_m],n_m} = 1 - \epsilon$
17 **return** x

Time Complexity Line 2 to 3 take $O(N)$. Considering the **while** loop from line 4 to line 7, every loop we have to check every channel to get the max payoff, so each loop takes $O(N)$. In the worst case, the final allocation could be $p_{L,n}, r_{L,n}$ for every $n < N$, and it takes NL loops to get that. The line 8 to the end takes constant time. So the complexity for the entire algorithm is $O(N^2L)$

TO DO: If we use a priority queue to store the $l[n]$. Every loop takes constant time. Then the complexity of the algorithm becomes $O(NL)$

Question 7.

TO DO After coding

Question 8.

DP Solution To find a DP Solution of the whole problem, we consider the subproblem: *find the best utility we can get using only the first n channels with total power limit* p' . Let $DP_{n,p}$ be the matrix who stores these values. We have the relation below:

$$
\begin{aligned}
&DP(0, p') = 0 \quad \forall p' \in \{0...p\} \\
&DP(n, 0) = 0 \quad \forall n \in \{0...N\} \\
&DP(n, p') = \max_{l \in \{0,\dots,L\}} \{DP(n-1, p' - p_{l,n}) + r_{l,n}\}
\end{aligned}
$$

By filling iteratively the matrix, we get the optimal allocation

DP-solution-Maximum-Utility(n)

```
DP = Zeros[N][p]
for q = 1 to p
    for n = 1 to N
        for l = 1 to L
            DP[n][q] = Max(DP[n − 1][q], DP[n][q - p_{l,n}] + r_{l,n})
return DP[N][P]
```

But this algorithm only return the maximum utility but not the optimal allocation. We need a way to store the optimal allocation and we decide to use another matrix of dimension (N, p) to store the optimal allocation for channel n in the sub problem (n, q)

DP-solution(n)

```
1   DP = Zeros[N][p]
2   LastTask = Zeros[N][p]
3   for q = 1 to p
4       for n = 1 to N
5           for l = 1 to L
6               l_M = 0
7               if ( DP[n-1][q - p_{l,n}] + r_{l,n} ) ≥ DP[n][q]
8                   DP[n][q] = DP[n-1][q - p_{l,n}] + r_{l,n}
9                   l_M = l
10          LastTask[n][q] = l_M
11  q = p
12  for n = N to 1
13      l_M = LastTask[n][q]
14      x_{l_M,n} = 1
15      q -= p_{l_M,n}
16  return x
```

Time Complexity The loop from line 3 to line 10 takes $O(NLp)$. The loop after line 12 takes $O(N)$. Thus the time complexity of the entire algorithm is $O(NLp)$

Space Requirement The matrices DP and LastTask take $O(Np)$

Question 9.

DP Solution Right now we consider the subproblem: *find the minimum power we need to reach total utility r using only first n channels.* Here r ranges from 0 to $U := \sum_{k=1}^{n} r_{L,k}$. U is the highest utility we can get from these n channels. Let $DP(n,r)$ be the matrix who stores these values. We have the relations below:

$$
\begin{aligned}
&DP(0,r) = \infty \;\; \forall r \in \{0...U\} \\
&DP(n,r^-) = 0 \;\; \forall n \in \{0...N\} \; \forall r^- \leq 0 \\
&DP(n,r) = \min_{l \in \{0,\dots,L\}} \{DP(n-1, r - r_{l,n}) + p_{l,n}\}
\end{aligned}
$$

By filling iteratively the matrix, we get the optimal allocation when $DP(N,r)$ reach p. If all values in $DP[N][:]$ are less than p, we can can reach the maximum utility U by assigning the maximum power.

```
DP-SOLUTION(n)
U = 0
for n = 1 to N
    U += r_{L,n}
DP = INFTY[N][r]
for n = 1 to N
    DP[n][0] = 0
LastTask = ZEROS[N][p]
for r = 1 to U
    for n = 1 to N
        for l = 1 to L
            l_M = 0
            if ( r - r_{l,n} ) ≥ 0 and ( DP[n-1][r - r_{l,n}] + p_{l,n} ) ≤ DP[n][r]
                DP[n][r] = DP[n-1][r - r_{l,n}] + p_{l,n}
                l_M = l
            elif ( r - r_{l,n} ) ≤ 0 and p_{l_n} ≤ DP[n][r]
                DP[n][r] = p_{l_n}
                l_M = l
        LastTask[n][q] = l_M
```

if DP$[N][r] >$ p
    $r_M = r - 1$
    break
$r = r_M$
for n = N to 1
    $l_M =$ LastTask$[n][r_M]$
    $x_{l_M,n} = 1$
    r -= $r_{l_M,n}$
    if $r \leq 0$
        break
return x

Question 10.

Branch-and-Bound We set each node as a subproblem defined as *The highest utility we can get with the constraint :* $[l_1^- \leq l_1 \leq l_1^+, ..., l_N^- \leq l_N \leq l_N^+)]$, where the tuple (l_n^-, l_n^+) represents the lower an upper bound of each channel n.

BB-SOLUTION(n)

$r_{max} = 0$;
$Q = deque()$;
PUSH$(Q, [(1, L), ..., (1, L)])$
while Q is not empty
    $[(l_1^-, l_1^+), ..., (1_N^-, l_N^+)] =$ PULL(Q)
    if $\sum_{k=1}^N p_{l_k^+,k} \leq p$
        if $\sum_{k=1}^N r_{l_k^+,k} \geq r_{max}$
            $r_{max} = \sum_{k=1}^N r_{l_k^+,k}$
            $A_{best} = [l_1^+, ..., l_N^+]$
            break
    for k = 1 to n
        if $l_k^- \leq l_k^+ - 1$
            break

$mid = floor(\frac{l_k^- + l_k^+}{2})$

PUSH(Q, $[(l_1^-, l_1^+), ..., (l_1^-, mid), ..., (1_N^-, l_N^+)]$

PUSH(Q, $[(l_1^-, l_1^+), ..., (mid + 1, l_k^+), ..., (1_N^-, l_N^+)]$

for n = 1 to N

$x_{A_{best}[n],n} = 1$

return x