Problem 1

Her fordeler vi først $\left\lfloor \frac{n}{P} \right\rfloor$ oppgaver til hver prosess, siden n ikke er delelig med P så er dette tallet mindre enn $\frac{n}{P}$ og vi står igjen med $n \mod P$ oppgaver å fordele. Vi gir 1 oppgave hver til $n \mod P$ forskjellige prosesser og fordelingen er da så lik som mulig.

Problem 2

Hvis vi nummerer oppgavene fra toppen nedover med 0 til og med 14 så kan vi lage en tabell for hvert tilfelle over hvilken oppgave som er blitt gjort av hver arbeider til hvilken tid. Vi antar at arbeiderene ikke blir slitne slik at vi må sirkulere hvem som gjør arbeidet og så videre,

tid (min)	arbeider 0	arbeider 1	arbeider 2	arbeider 3
10	0			
20	1	2		
30	3	4	5	6
40	7	8	9	10
50	11	12	13	14

For fire arbeidere er altså svaret 50 minutt. Hvis vi gjentar prosedyren for tre arbeidere, blir svaret da 60 minutt.

Problem 3

Først introduserer vi en blokkfordeling, en blokkfordeling er en fordeling hvor arbeidet blir fordelt i store blokker som er så like mulig (a la Problem 1). Hvis vi ser for oss et flatt array A med n elementer som skal fordeles til R prosesser så får en prosess med rang k elementer fra en blokk av A med laveste indeks $L(k, R, n) = \left\lfloor \frac{k \cdot n}{R} \right\rfloor$, høyeste (ikke inklusiv, altså elementet på denne indeksen tilhører neste prosess) indeks $H(k, R, n) = L(k + 1, R, n)$, størrelsen på blokken blir da $S(k, R, n) = H(k, R, n) - L(k, R, n)$.

For å fordele griddet så jevnt som mulig så skal vi blokkfordele de M kolonnene i x retning opp i P blokker, og så blokkfordele de fordelte kolonnene med lenge N i Q blokker.

Så hvis vi har en prosess R_k med rang k så må vi først lage en tilsvarende 2d rang (i_k, j_k) som bestemmer hvor i griddet den skal ligge, vi legger de P første prosessene i øverste prosess rad også etter det ligger de under hverandre, med dette i tankene så ser vi at $i_k = k \mod P$ og $j_k = \left\lfloor \frac{k}{P} \right\rfloor$ passer siden første rang i_k blir da lik for alle prosesser i en prosesskolonne og andre rang j_k blir lik for alle prosesser i en prosessrad. Fordelingen vil altså se omtrent ut som dette,

$R_0 = R_{0,0}$	$\cdots$	$R_{P-1} = R_{P-1,0}$
$R_P = R_{0,1}$	$\cdots$	$R_{2P-1} = R_{P-1,1}$
$\vdots$	$\ddots$	$\vdots$
$R_{PQ-P} = R_{0,Q-1}$	$\cdots$	$R_{PQ-1} = R_{P-1,Q-1}$

Så en prosess med rang k blir altså tildelt den delen av griddet som begynner med indeks $L(i_k, P, M)$ i x retning og indeks $L(j_k, Q, N)$ i y retning med lengde $S(i_k, P, M)$ i x retning og lengde $S(j_k, Q, N)$ i y retning.

En ting som er viktig å ta med her er at prosesser som er naboer i x retning må være enige om den lokale størrelsen til kolonnene slik at de enkelt kan kommunisere hele kolonner til hverandre, det er derfor også viktig for prosesser som er naboer i y retning om å være enige om den lokale størrelsen til radene, disse egenskapene blir ivaretatt fordi j_k er lik for prosesser som er naboer i x retning og i_k er lik for prosesser som er naboer i y retning slik at $S(j_k, Q, N)$ og $S(i_k, P, M)$ er like for disse naboene.

Problem 4

Vi lar $T_m(n)$ være tiden det tar en m-stage pipeline å prosessere n oppgaver. For å få oversikt så lager vi en tabell over hva som skjer i en m-stage pipeline som prosesserer n oppgaver $(t_0, t_1, \cdots, t_{m-1})$, oppgavene er delt opp i m deler igjen $(t_k^0, t_k^1, \cdots, t_k^{m-1})$,

$$
\begin{array}{c|c|c|c}
t_0^0 & & & \\
t_0^1 & t_1^0 & & \\
\vdots & \vdots & \ddots & \\
t_0^{n-1} & t_1^{n-2} & \cdots & t_{n-1}^0 \\
t_0^n & t_1^{n-1} & \cdots & t_{n-1}^1 \\
\vdots & \vdots & \vdots & \vdots \\
t_0^{m-1} & t_1^{m-2} & \cdots & t_{n-1}^{m-n-2} \\
 & t_1^{m-1} & \cdots & t_{n-1}^{m-n-1} \\
 & \ddots & \vdots & \vdots \\
 & & t_{n-2}^{m-1} & t_{n-1}^{m-2} \\
 & & & t_{n-1}^{m-1}
\end{array}
$$

Vi ser at vi først får en seksjon hvor en oppgave blir påbegynnt om gangen opp til punktet vi får full utnyttelse av m-stage pipelinen og til slått får en seksjon hvor en oppgave blir avsluttet om gangen. Vi forsøker å telle tiden til den første seksjonen, for å påbegynne $n-1$ (rett før vi får full utnyttelse av hele pipelinen) oppgaver tar det $n-1$ tidsenheter, tiden det tar å avslutte alle oppgavene er åpenbart like lang som den første seksjonen, så vi har nå telt $2(n-1)$ tidsenheter. Hvis vi teller delen der vi har full utnyttelse av pipelinen så får vi $m - n + 1$ tidsenheter, vi har nå totalt $2(n-1) + m - n - 1 = 2n - 2 + m - n + 1 = m + n - 1$ tidsenheter.

Vi danner nå en hypotese, vi antar at $T_m(k) = m + k - 1$ for en eller annen $k \in \mathbb{N}$, og forsøker å vise at dette impliserer at $T_m(k+1) = m+k$ eller tilsvarende $T_m(k+1) = T_m(k) + 1$. Vi ser lett at dette stemmer for alle $k < m$ fordi vi ikke har oppnådd full utnyttelse av m-stage pipelinen for disse verdiene slik at vi bare får et nytt steg i delene hvor oppgavene starter og stopper en om gangen, hvis $k = m$ så har vi ett steg hvor vi har full utnyttelse av pipelinen, men vi ser også at vi kan legge en ny oppgave (den $k+1$'te oppgaven) rett etter t_0 slutter, vi ender da praktisk talt opp med å bare legge til et nytt steg i den delen som slutter og påstanden holder også her, det samme kan vi gjøre for $k > m$.

Det som står igjen nå er å teste om basishendelsen $T_m(1)$ stemmer, vi vet at å utføre en oppgave med m deler må skje i m tidssteg siden vi ikke har noen mulighet for å parallelisere, vi får altså at $T_m(1) = m$ som stemmer med $T_m(1) = m + 1 - 1$. Vi har altså vist for alle $n \in \mathbb{N}$ at tiden det tar å utføre n oppgaver i en m-stage pipeline er $T_m(n) = m + n - 1$.

Problem 5

Etter notasjonen i forrige oppgave får vi $T_{\text{pipe}} = T_m(n)$ og $T_{\text{seq}} = m \cdot n$, definisjonen av speedup er $\Psi = \frac{T_{\text{seq}}}{T_{\text{pipe}}}$. Vi setter opp problemet og løser mht. til n,

$$\begin{aligned}
\frac{m \cdot n}{m + n - 1} &\geq p \\
m \cdot n &\geq pm + pn - p \\
mn - pn &\geq pm - p \\
n(m - p) &\geq pm - p \\
n &\geq \frac{p(m-1)}{m-p}.
\end{aligned}$$

Her gjør vi antakelsen at $m - p$ er større enn 0 som er rimelig da vi kan umulig få større speedup enn vi har m enheter i pipelinen.