Give a precise definition of a subproblem that can be used to solve the SpacedSum problem with a dynamic programming algorithm.

Solution. The subproblem is - largest possible value that can be obtained by summing a subset of the entries of A with indices from 1 to i satisfying the constraints. Denote $S(i)$ this value.

Describe and justify the recurrence rule that will be used to compute the solutions to the subproblems defined in part (ii).

Solution. Correctness: For $S(i)$ there are only two possibilities:
- if we use the entry $A[i]$ then $S(i) = S(i-3) + A[i]$, as we must skip entries $A[i-1]$ and $A[i-2]$;
- if we do not use the entry $A[i]$ then $S(i)$ is the same as $S(i-1)$.

The largest of the solutions to the subproblems gives the value of the optimal solution.

The base cases are:
$n = 0; S(0) = 0;$
$n = 1; S(1) = A[1];$
$n = 2; S[2] = \max(A[1], A[2]).$

The DP recurrence is

$$S(i) = \begin{cases} 0 & \text{if } i = 0 \\ A[1] & \text{if } i = 1 \\ \max(A[1], A[2]) & \text{if } i = 2 \\ \max(S(i-3) + A[i], S(i-1)) & \text{if } i > 2 \end{cases}$$

Provide the pseudocode for your dynamic programming algorithm that solves the SpacedSum problem using the subproblems and recurrence rule defined above.

Solution.

```
S[0] = 0; S[1] = A[1]; S[2] = max(A[1],A[2]);
for i from 3 to n do
  S[i] = max(S[i-3]+A[i], S[i-1])
od
RETURN S[n]
```

) Analyze the time complexity of the algorithm obtained in part (iv).

Solution. Loop iterates $n-2$ times with every iteration taking constant time. $\Theta(n)$