Write your answers in the separate answer booklet.

1. A *data stream* is an extremely long sequence of items that you can only read only once, in order. A good example of a data stream is the sequence of packets that pass through a router. Data stream algorithms must process each item in the stream quickly, using very little memory; there is simply too much data to store, and it arrives too quickly for any complex computations. Every data stream algorithm looks roughly like this:

```
DoSomethingInteresting(stream S):
  repeat
      x ← next item in S
      ⟨⟨do something fast with x⟩⟩
  until S ends
  return ⟨⟨something⟩⟩
```

   Describe and analyze an algorithm that chooses one element uniformly at random from a data stream, *without knowing the length of the stream in advance*. Your algorithm should spend $O(1)$ time per stream element and use $O(1)$ space (not counting the stream itself). Assume you have a subroutine RANDOM(n) that returns a random integer between 1 and n, each with equal probability, given any integer n as input.

2. Consider a random walk on a path with vertices numbered $1, 2, \ldots, n$ from left to right. We start at vertex 1. At each step, we flip a coin to decide which direction to walk, moving one step left or one step right with equal probability. The random walk ends when we fall off one end of the path, either by moving left from vertex 1 or by moving right from vertex n.

   Prove that the probability that the walk ends by falling off the *left* end of the path is exactly $n/(n+1)$. *[Hint: Set up a recurrence and verify that $n/(n+1)$ satisfies it.]*

3. Consider the following algorithms for maintaining a family of disjoint sets. The UNION algorithm uses a heuristic called *union by size*.

```
MakeSet(x):
  parent(x) ← x
  size(x) ← 1
```

```
Find(x):
  while x ≠ parent(x)
      x ← parent(x)
  return x
```

```
Union(x, y):
  x̄ ← Find(x)
  ȳ ← Find(y)
  if size(x̄) < size(ȳ)
      parent(x̄) ← ȳ
      size(x̄) ← size(x̄) + size(ȳ)
  else
      parent(ȳ) ← x̄
      size(ȳ) ← size(x̄) + size(ȳ)
```

   Prove that if we use union by size, FIND(x) runs in $O(\log n)$ time *in the worst case*, where n is the size of the set containing element x.

4. Recall the SUBSETSUM problem: Given a set X of integers and an integer k, does X have a subset whose elements sum to k?

   (a) **[7 pts]** Describe and analyze an algorithm that solves SUBSETSUM in time $O(nk)$.

   (b) **[3 pts]** SUBSETSUM is NP-hard. Does part (a) imply that P=NP? Justify your answer.

5. Suppose we want to maintain a set X of numbers under the following operations:

   - INSERT(x): Add x to the set X.
   - PRINTANDDELETEBETWEEN(a, z): Print every element $x \in X$ such that $a \leq x \leq z$, in order from smallest to largest, and then delete those elements from X.

   For example, PRINTANDDELETEBETWEEN($-\infty, \infty$) prints all the elements of X in sorted order and then deletes everything.

   (a) **[6 pts]** Describe and analyze a data structure that supports these two operations, each in $O(\log n)$ *amortized* time, where n is the maximum number of elements in X.

   (b) **[2 pts]** What is the running time of your INSERT algorithm in the worst case?

   (c) **[2 pts]** What is the running time of your PRINTANDDELETEBETWEEN algorithm in the worst case?