University of Nevada, Las Vegas Computer Science 477/677 Spring 2022

Answers to Assignment 6: Due Wednesday April 13, 2022, midnight.

1. Your company sells boating accessories. You need to choose a hash function for your set of customer files. Which of the following would be a better hash function?

   (a) The last four digits of the customer's social security number.

   (b) the last four digits of the customer's zip code.

   Why did you make that choice?

   The last four digits of the social security number. The zip code is correlated with the customers location, which is heavily weighted toward areas near water.

2. If separate chaining is used to resolve collisions in a hash table of size m which stores n items, the probability that a given place has exactly k items is approximately

$$\frac{(n/m)^k}{k!\, e^{n/m}}$$

   as given by the Poisson distribution. If $m = 100$ and $n = 200$, the average number of items in a place is 2. Approximately how many places will have exactly 2 items? Choose from the following list.
   100
   55
   45
   27
   21
   13

   For this example, $n/m = 2$, and $k = 2$. The probability that any given place has exactly 2 data items is approximately $\frac{2^2}{2!e^2} = \frac{2}{e^2} \approx 0.27067$. There are 100 places, and thus the expected number of places that have exactly 2 data items is approximately 27.

3. You are trying to construct a cuckoo hash table of size 10, where each of the 9 names listed below has the two possible hash values, indicated in the array. Can you construct that table? Construct the table, or show that it can't be done by using Hall's marriage theorem.

You must show the items that are ejected as strikeouts.

	h1	h2
Ann	0	3
Bob	1	3
Ted	6	8
Sue	3	6
Gus	2	7
Cal	4	7
Dan	1	9
Sal	6	9
Eve	5	8

0	Ann
1	~~Bob~~ ~~Dan~~ Bob
2	Gus
3	~~Sue~~ ~~Bob~~ Sue
4	Cal
5	Eve
6	~~Ted~~ ~~Sue~~ Sal
7	
8	Ted
9	Dan

4. A 3-dimensional $9 \times 7 \times 10$ rectangular array A is stored in main memory in row major order, and its base address is 8192. Each item of A takes one word of main memory, that is, one addressed location. Find the address of A[4][5][2].

   We need to compute the offset of A[4][5][2]. The predecessors of that item consist of a 3-dimensional (3D) block, a 2D block, and a 1D block. The 3D block consists of items A[0][0][0] through A[3][6][9], $4 \times 7 \times 10 = 280$ entries. The 2D block consists items A[4][0][0] through A[4][4][10], $1 \times 5 \times 10 = 50$ items. The 1D block consists of items A[4][5][0] through A[4][5][1], $1 \times 1 \times 2 = 2$ items. Thus there are $280 + 50 + 2 = 332$ predecessors.

5. Explain how to implement a sparse array using a search structure. (This **exact** problem will be on the next examination on April 20.) Hint: You must explicitly describe how to implement the operators **fetch** and **store**.

   A sparse array A has a default value, which could in theory be anything, but is usually zero, NULL, or blank. We'll assume it's zero. Let S be a search structure. (Do not specify what kind of search structure, since that is not the point of this question.)

   The items stored in S are ordered pairs of the form (i, x), meaning that $A[i] = x$.

   To implement **store**(A, i, x), the assignment $A[i] = x$:
   Execute Find(i)
   If the return value is (i, y), replace that pair by (i, x) in S.
   If Find fails, insert the pair (i, x) into S.

   To implement **fetch**(A, i):
   Execute Find(i)
   If the ordered pair (i, x) is found, **fetch** returns x.
   Otherwise, return 0.

6. Suppose you wish to store the values $\binom{n}{k}$ for all $n \leq N$ for some constant N. Recall that $0 \leq k \leq n$. These values form a triangular array, shown below for $N = 6$. To save space, you will store the values in an 1-dimensional array $A[M]$, where M is the size of the triangle. For example, $M = 21$ if $N = 6$. Write a function to fetch values. Here is your function, with one missing formula.

```
int choose(int n, int k)
 {
  int index = n*(n+1)/2 + k;
  return A[index];
 }
```

Here is Pascal's triangle for $n \leq 6$.

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5  10  10   5   1
1   6  15  20  15   6   1
```

7. Write, in pseudocode, an algorithm which writes a maximum total subsequence of a sequence of positive numbers, with the rule that no two consecutive terms of the sequence may both be selected. Your algorithm should use dynamic programming.

Hint: The i^{th} subproblem is to find the maximum total of any subsequence which uses only the first i terms of the original sequence.

Let $x[i]$ be the i^{th} term of the sequence. Let the indices start at 1. Define $S[i]$ to be the maximum total of any legal subsequence of $x[1], \ldots x[i]$.

Here is the pseudocode.
$S[0] = 0$
$S[1] = x[1]$
For all i from 2 to n
  $S[i] = max(S[i-1], S[i-2] + x[i])$
Return $S[n]$.

The following recursive code prints the maximum total legal subsequence.

```
void writesubsequence(int i)
 {
  if(i > 0)
   if(S[i] > S[i-1])
    {
     writesubsequence(i-2)
     cout << " " << x[i];
    }
```

```
   else writesubsequence(i-1);
 }

int main()
 {
  writesubsequence(n);
  cout << endl;
  return 1;
 }
```