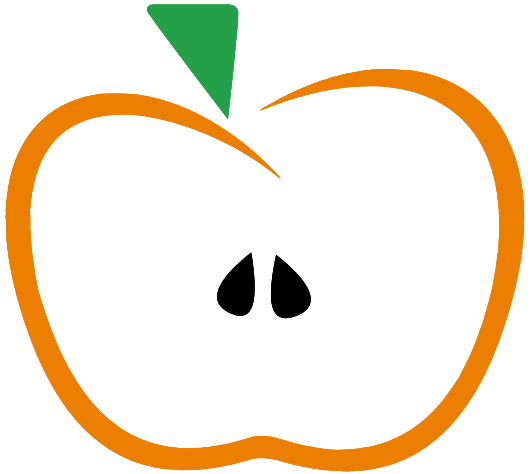
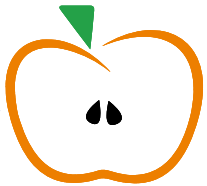
Getting Work Done with APL

Josh David



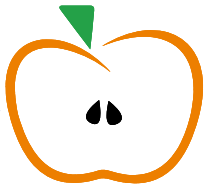
APL Seeds

- What does it look like when my seeds come to fruition?
  - is it worth the investment?
- What do seeds fundamentally need to thrive?
  - Soil, water, light



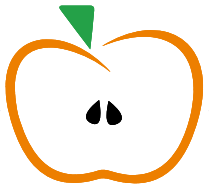
Role of an APL Interpreter

- Boil a problem down into simple arrays, exposing Iverson toolkit



System Functions for File Read/Write

- NREAD
  - …
- FREAD
  - …
- CSV
- XML
- JSON
- MAP



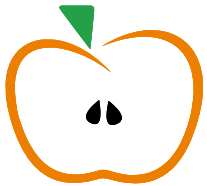
Water Potability

Property	Description
pH	PH is an important parameter in evaluating the acid–base balance of water. It is also the indicator of acidic or alkaline condition of water status.
Hardness	Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geologic deposits through which water travels.
Solids	Water has the ability to dissolve a wide range of inorganic and some organic minerals or salts such as potassium, calcium, sodium, bicarbonates, chlorides, magnesium, sulfates etc.
Chloramines	Chlorine and chloramine are the major disinfectants used in public water systems.
Sulfate	Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. They are present in ambient air, groundwater, plants, and food.
Conductivity	Pure water is not a good conductor of electric current rather's a good insulator. Increase in ions concentration enhances the electrical conductivity of water.
Organic Carbon	Total Organic Carbon (TOC) in source waters comes from decaying natural organic matter (NOM) as well as synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water.
Trihalomethanes	THMs are chemicals which may be found in water treated with chlorine.
Turbidity	The turbidity of water depends on the quantity of solid matter present in the suspended state. It is a measure of light emitting properties of water.

Water Potability

```
l<(¨1)m                 For each row of this matrix
                                For each item of this vector of limits
                                        Compare against its respective column
```

```
(l<(¨1)m)^(u>(¨1)m)     For each bound in upper and lower
                                For each row of this matrix
                                        For each item of this vector of limits
                                                Compare against its respective column
                        and then AND each matrix together..
```



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mzXML
 xmlns="http://sashimi.sourceforge.net/schema_revision/mzXML_2.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://sashimi.sourceforge.net/schema_revision/mzXML_2.0 http://sashimi.sourceforge.net/schema_revision/mzXML_2.0/mzXML_idx_2.0.xsd">
 <msRun scanCount="7161"
        startTime="PT0.00683333S"
        endTime="PT200.036S">
  <parentFile fileName="file://Rdf3/data2/search/ppatrick/sashimi_repository/LCQ/7MIX_STD_110802_1.RAW"
              fileType="RAWData"
              fileSha1="957f3baf650d4de3d87c04a9fc64baa13f6b363e"/>
  <msInstrument>
  <msManufacturer category="msManufacturer" value="ThermoFinnigan"/>
    <msModel category="msModel" value="LCQ Deca XP"/>
  <msIonisation category="msIonisation" value="ESI"/>
  <msMassAnalyzer category="msMassAnalyzer" value="Ion Trap"/>
  <msDetector category="msDetector" value="EMT"/>
    <software type="acquisition"
              name="Xcalibur"
              version="1.3 SP 1"/>
  </msInstrument>
  <dataProcessing centroided="1">
    <software type="conversion"
              name="Thermo2mzXML"
         version="1"/>
  </dataProcessing>
  <scan num="1"
        msLevel="1"
        peaksCount="705"
        polarity="+"
        retentionTime="PT0.41S"
        lowMz="400"
        highMz="1500"
  basePeakMz="1221.89"
  basePeakIntensity="6.0092e+006"
  totIonCurrent="2.69539e+008">
    <peaks precision="32"
           byteOrder="network"
           pairOrder="m/z-int">Q8hAfEkweKBDyMyyR7tyAEPJpA5JiZHIQ8o9hkkUZKBDyu8sSPZTQEPLY+hJNOsQQ8vQaEk7emBDzE+QSO9nYEPMyUhJBpLgQ81C4ElpvGBDzcJQSRCu8EPOLF5H7LWAQ86lUElnNDBDzwFMSH02QEPPjw5HPo4AQ8/uekj1rABD0Fpy
  <scan num="2"
        msLevel="2"
        peaksCount="74"
        polarity="+"
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mzXML
 xmlns="http://sashimi.sourceforge.net/schema_revision/mzXML_2.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://sashimi.sourceforge.net/schema_revision/mzXML_2.0 http://sashimi.sourceforge.net/schema_revision/mzXML_2.0/mzXML_idx_2.0.xsd">
 <msRun scanCount="7161"
        startTime="PT0.00683333S"
        endTime="PT200.036S">
  <parentFile fileName="file://Rdf3/data2/search/ppatrick/sashimi_repository/LCQ/7MIX_STD_110802_1.RAW"
              fileType="RAWData"
              fileSha1="957f3baf650d4de3d87c04a9fc64baa13f6b363e"/>
  <msInstrument>
  <msManufacturer category="msManufacturer" value="ThermoFinnigan"/>
    <msModel category="msModel" value="LCQ Deca XP"/>
  <msIonisation category="msIonisation" value="ESI"/>
  <msMassAnalyzer category="msMassAnalyzer" value="Ion Trap"/>
  <msDetector category="msDetector" value="EMT"/>
    <software type="acquisition"
              name="Xcalibur"
              version="1.3 SP 1"/>
  </msInstrument>
  <dataProcessing centroided="1">
    <software type="conversion"
              name="Thermo2mzXML"
         version="1"/>
  </dataProcessing>
  <scan num="1"
        msLevel="1"
        peaksCount="705"
        polarity="+"
        retentionTime="PT0.41S"
        lowMz="400"
        highMz="1500"
  basePeakMz="1221.89"
  basePeakIntensity="6.0092e+006"
  totIonCurrent="2.69539e+008">
    <peaks precision="32"
           byteOrder="network"
           pairOrder="m/z-int">Q8hAfEkweKBDyMyyR7tyAEPJpA5JiZHIQ8o9hkkUZKBDyu8sSPZTQEPLY+hJNOsQQ8vQaEk7emBDzE+QSO9nYEPMyUhJBpLgQ81C4ElpvGBDzcJQSRCu8EPOLF5H7LWAQ86lUElnNDBDzwFMSH02QEPPjw5HPo4AQ8/uekj1rABD0Fpy
  <scan num="2"
        msLevel="2"
        peaksCount="74"
        polarity="+"
```

Q8hAfEkweKBDyMyyR7tyAEPJpA5JiZHIQ8o9hkkUZKBDyu8sSPZTQEPLY+hJNOsQQ8vQaEk7emBDzE+QSO9nYEPMyUhJBpLgQ81C4ElpvGBDzcJQSRCu8EPOLF5H7LWAQ86lUElnNDBDzwFMSH02QEPPjw5HPo4AQ8/uekj1rABD0FpySEulw
EPQ8vpIaOcAQ9Gw9klOdSBD0jHmSV/DAEPSh25H6xKAQ9LoXEfz1IBD02l4R77yAEPUKcJI7RfAQ9SRED+AAABD1Q5KSCtlgEPVwxpILEcAQ9Y2TEjzcsBD1stQSQNosEPXGHpHh46AQ9d6iEeJh4BD19vmSO2rgEPYbBpItR5gQ9jJxkj+LSBD2T
RCSBF+gEPZgYpHqEEAQ9oLQEidDMBD2mxOSL1hQEPbA7pIH7tAQ9uHAki6SwBD3AOKSVLtkEPc9bhJJK+AQ92GuklAFzBD3g/ASIApIEPemc5J7C/oQ98cXElBXtBD37eQSSLN0EPgYkZJi9ZYQ+FVHEl0BwBD4fsISU6lIEPiuBZJJhnQQ+NGckj
dkMBD5FDMSRtjMEPk49BIlNngQ+WPXEiCxaBD5loUR/7rAEPm+IxJlg6wQ+desElonoBD5/mYSOR44EPonWpJEcfgQ+kCQkiiPkBD6bw8SQMQcEPqOtRJEXuwQ+q/ZkhdPQBD6xOUR8/8AEPrcypIU+FAQ+vd+EkDwcBD7HFsSUSX0EPs2/JI3F
XAQ+0zBEhFToBD7iNgSUw7EEPufnpIoaBAQ+8ajElP++BD73wgSKNoYEPwRCJISumAQ/C2QkhqzIBD8TiUSPLHYEPxxdhIDXLAQ/JVwEkCLhBD8vZkR/7qAEPzRRZHRG4AQ/RV3khvX8BD9MI6R6X5AEP1f4xJerwgQ/XbDkMJAABD9jkySMk54
EP3B6pIomYAQ/euvEiCFcBD+JtySJYMQEP4/nZCHAAAQ/pd/EkBigBD+w7oSNPzQEP7Z3hHRU4AQ/vk2EkIczBD/HzUSRip8EP80f5Ik8fgQ/1ZJkhW4EBD/gwiSUVdUEP+nMZIU0mAQ/8EPEj1yaBD/2lIR4UmgEQAAZ5IhYDARAAorEf1zABEAF
WQSXjy8EQAjIBCLAAARADpEhduYBEA8ESN3SQEQBPlBHnZSARAF9QEifuUBEAbmgSki0LEQB99BJjOC4RAI3YEm1RhBEAoP4SafuQEQC1lZIpaWARAMMjEiPmUBEA1sCSQF5AEQDmbxI1OuARAPBdEkT+7BEA+jcR1mKAEQEE85I9+lARAQ
9WEKKAABEBLEUSLcrYEQE1/hIOY/ARAU4oEkQaTBEBXV2SXk0UEQFsFpBEAAARAX7dEnppHhEBj9cShsQHEQGjMxJCISwRAbd7Ee2OwBEBxMqSL2cQEQHeuxIZk6ARAe54Eg4noBECCzCSHktgEQIj6RIV4JARAj2NEj4pSBECV3aSS/JAEQJwH
xJVgdQRAn5FEjChKBEClpcR6uhgEQKmWRH+seARArLzkinygBECwX6SGvSQEQLhnhJdYMQRAu0ZkjJ9GBEC9vcSOzG4EQMHy5ID4bARAxTnEgVPYBEDOAkSMOaoEQNOsRJPmHwRA1u7j+AAABEDc5WSIr1wEQOAGRIckRARA6SUEjnKWBED
tQESFZcgEQO+txHosWARA+AfEkhQsBED9X0SR2UcEQQIWxH+vuARBCSLkigP4BEEPPASJjRwEQRKQhIn8/ARBFSAEcefgBEEYhYRmxMAEQSFZxJl9DIRBI9HEIIAABEEnjER/jQAEQTGWhHkN8ARBPgmkdUSQBEFCnASXTR4EQUUhRHxRcARB
SANkmStvhEFLpIScXiUEQVCDBIg4+ARBV26EiKwQBEFgKcSYq2QEQWQrBJYUhwRBaZpEhlnoBEFz4kSMYiAEQXgRpIdiAARBgCNEhn38BEGHqwSahIAEQYw9hI8/xgRBj+MEjP6OBEGUCoSSbj8EQZhlxIycLgRBnQVElekwBEGnYeSUUckEQatk
RIs+3ARBsO/kjOcmBEG0R4SJsZIEQbhvRIaE+ARBvHAEiHk0BEHC40SHXVQEQcWihCmAAARByDfkiJQgBEHLXcSIK34EQdNuhG9VIARB2UkEjVMGBEHdxwSM7UwEQeLgRIu8CgRB6GJEiyhSBEHv0SSUlZoEQfhPhInvoARB+zJEj0q8BEH/WES
Dg6AEQgJNRIDiDARCDQxEhbeEBEIPdQR4SCgEQhWa5IsdDgRCGTsEiyGoBEIcREQAAAAEQiEXRIzcWgRCJBLEjBiGBEIm5cRv1wAEQiuhJImXGgRCLu8EjPLuBEIxowQmgAAEQjY2xI81igRCOoqEgkN0BEJAAiSK6ZwEQkLRxI0j2gRCTlxkkLnGBE
JTi4SIZ/gEQlt8hIA9oARCZXREib6aBEJqa0SOYSAEQm8JBJKjYwRCdBzEjlPgBEJ3GESIoQgEQnwOxH4lwARCjOlEiQGqBEKX6mSODSoEQpuXZHn8mARCoTyEkWU BBEKsK8SKNVoEQrD2xHU+sARCtgPEmLfuBEK8C8SLzAwEQsDHxIL5UARCx
h5kjLd2BELKcESGFBwEQtBYhIusQARC0sNkKgAABELd6UR9YygEQuO6RJCHHARC6xDkjE1aBELyCUSW7OcEQvTpxDUQAARC+UuEiTNsBEL/P6SDoCwEQwYABJS/sQRDC6lEgft0BEMOz4SHNTAEQxdMpIxUcARDGf6EbiTgBEMtpYSMQtYEQz
AMxHnqQARDOR/kf1X4BEND7qSI97oEQ0b/RIE6eARDSnWEkLr6BENRmoSQht0EQ1WDxI5xYgRDWPtEhS98BENczMR8X8gEQ3e5BJNzGgRDeuQEkVORBEN/QCR98VAEQ4O2JIeU9ARDhqHki8YABEOJZ+QTAAAEQ5F+RIzC2gRDmBHkj0pg
BEOa4uR9PDAEQ55MZH9rsARDo1jEf9roBEOs4oSEU1QEQ7djhIjNegRDxv5EjXayBEPLzKSNimwEQ9DE5IsM7gRD01IEJUAABEPbsKSDA6gEQ970xIh6dARD5AjEhoacBEQHX0SRxBEERAqABBAAAAREDs7EjqNgBEQSEISUQucERBVAxBoAAAR
EGM0EjSkuBEQblkQfgAAERB87xJmC8oREIyHEk5AcBEQxSQSRhmEERDVjRHsweAREQwlEhycwBERIq0SPODAEREw5xIidsARETz2EgXi8BERURESKkSAERFevxHxCIAREX1vEiZaWBERk2mSBN8QERHWehIsSjAREeSOkEwAABER+08SGkTwERI
VEBI66GAREjLCkjX7oBESRfqSDhhAERJgRH0T8AREm/1kiqUUBESh4ASMntwERKi5BIhdPARErwEEg1RcBES0gsSHaegERLkiBIa7lAREu+Nkce1wBES/pESTb+kERMMFpHFpoAREyPJEeFDoBETWMYR7ibAERODHxH4h2ARE5HaEdI+QBEToE
YR55YAERO3IBI7RAARE96cEiAsqBET7x2SJjuoERQWwhIIvPARFCXCkgaA8BEUQRkSGRLAERROXhIhVsARFF20EkHFEBEUZ5IQiQAAERSJVBJE32ARFJdhEeSIIBEU0v0SNHGIERT0oxHydmARFQPKEjTH0BEVLwYR5bDAERU+p5IEE6ARFVDqE
gU00BEViq6SLP7wERXMJhH+RkARFebiEghNgBEWBY0R+5sgERYRcxIVg6ARFirmEidtsBEWTOoSBFwAERZXqJIrPgARFoT/EitlkBEWnZ0SKB4QERaqRJI5lOgRFrSnEfoogBEWxgcSH+FgERby3ZIQBBARFxh4EhPPcBEXI+kQEAAAERc6hhJBln
wRF1OSEjrTQBEXXyUQKAAAERdq6hIrM2gRF3hMEh1v8BEXj+MSN/d4ERecRBAgAAARF6/ykifi6BEXv2sR+s8gERfY2RI787gRF+vFEjx6iBEYGw8SEc5gERgy5xHzB2ARGEYuEidK4BEYVvER8CEgERhjMRIEdLARGHZbkjyjOBEYm/gSEhUgERjE
JhHT5QARGNauEjsNQBEY6yKSBlRgERj/ZpJDLngRGSCNkhcHgBEZQm0SQ08MERlYo5IYSLARGWMdEdxwgBEZdEcSPsJIERl+7pAoAAARGZhsEkB/iBEZu1kSJxvQERntohIPRgARGgLRkhQnkBEaQDqSNPVgERpMzI/gAAARGmKvkhrP8BEaenc
SJjvAERqEqRDTQAARGpS3khHXkBEazFsSTw1oERr8MZIm+SARGwY4Ei2w2BEbFgeSNDTAERsmjhIhz2ARG0EUEhAQUBEbafiSBrSAERt9I5JC68gRG4egEgJ34BEbo2SSMJqoERu0DZHuNyARG9qJEikymBEb8N2SRDToERwB+JH5mWARHBP
KkjkJMBEcKDUSJGzIERw4PBHzIKARHFtAkiETKBEcdCASHMKgERy/QpH/wAARHOUdkit5oBEdNlYSEbQgER1RXhIhdRARHWDOkhaFkBEdbmeSG+cwER18IxHSbYARHZV0EkrDOBEdoglSH43gER29RRIU2IARHc5JEiGIOBEeHRKSBHzwER46w
BILYDARHk5pkcTrwBEecX4RwHnAER6sGhIgHJARHtPEkiG72BEe3xMSJQpIER8C65IXOoARHw9FkiNE+BEfG1USAyvgER9IgBHZFgARH1y7kfjuYBEfalISLGgwER9+cRHjQoARH5pUEjd62BEftmUR5BUAER/AOBISPcARH9xZkmoqqBEf6UaSX
QrUER/2IxIuuZgRIADZEjl4EBEgD6ESKKjgESAY6xI1UogRICCuEixd4BEgOx0SRWMQESBCiBIfVPARIFMoEikyUBEgWRYR3NKAESBisRlwdygRIG0tEjUG0BEgcokR40+gESB7JxIvwpgRIlqBEgsxIBEglqcSI0zIESCqiRIbm9ARILsdkkVY7BEgwheSI5
EgESDKeRIaxqARINn8EffAoBEg/60SE5ewESEI0BIv0kARIRCWEfeogBEhItkSMmJgESEr65IrJSARITHhkfWIIBEhW8KSTNEMESFhcpILODARIXuxEfsBoBEhh5aSJHeAESGPiRIN6CARIZ44EkAs6BEhpqiSNG14ESHSYxIjPQARIdpEEj35+BEh5ikSH
2pAESH+HRI3VbARIgNsEMSAABEiDtgSKLz4ESIVJ5AQAAARIi/TkkQnBBEiNWIQgAAAESJBvRJMjJARIlrkEhx7sBEijMMR6UWgESKTrxIuWNgRIpyeEhn+EBEisNQSDkxwESLIdRHzukARIs72kgAX8BEi2SUSKv7YESLexRC0AAARIuuukhHikBEi//k
SCf3wESMGjhIuw4ARIxAsEqfqK5EjGBISaT38ESMd0xIRG5ARIyLukle90BEjKiMSJMpAESM5ORHLf0ARI0TBEjN8oBEjTg8R2DvAESNpWxILYBARI4afkkGRDBEjjGkQTAAAESOaKZJBS9ARI6hnkjFN+BEjt+sSLTLgESO/fhIL+FARI8R2Eih+mBEj0l
YSFQpwESPsFRINL5ARJAn+EgvpIBEkErYSLDugESQbRBIlcMARJCSiEhU+IBEkOe0SCx2QESRDoRHnaCARJFa+khejYBEkcSISKU8gESR8vRJTM9ARJIRmECAAABEkj+kSJfwgESSyQxIUZgARJLjHEgCg8BEkzAYSFKXAESTSlRImlsgRJNiJkgWTw
BEk5YMSIAMoESTzfhIQ9AARJQFfkeOrIBElCS8SI1xYESUT2BJAHNQRJRk/EJ4AABElL1KSR18gESU2QBI0XBgRJT6cElISSBElRFySK3JwESVKyJIrUuARJVHOEihOSBElW8qSKI2QESVhJRIjqFARJXHHkhOi0BEljgoSVZrcESWUXxIGKyARJaNyEk
HWnBElunYSIk3AESXSPxJNeNgRJdiBEji+CBEl3+cSPEQgESX2CpH1+aARJfx8EkmclBEmBfsSKcx4ESYaGhI13LgRJh+6EhuF0BEmJnGSHvswESYvHpKt2LkRJji2Ei0rIBEmRD4SZC8uESZmxxINfCARJnI6kgrwIBEmd34SKIiIESaFLBIinIgRJoxCE
eiMABEmk9mR7bTgESaY1pJDfPQRJtAdElOT8BEm3XESc894ESbrcpJWQsQRJyYaEiRp0BEnNEMRzq3AESdL5hIcqaARJ1PaEgEkABEnXqYSMS1YESdkTBJC4XARJ2/GEdW0ABEneecSHkeAESeHyxID0cARJ6t6EiQ/gBEnvMOSM7EQESfPxxI5
BpgRJ+CikfuYwBEn99cSFzlAESgGxRHcI8ARKCz8Egl4YBEoMd8SKhYYESg9NJIyATARKFFOEcjjQBEoYtmSWQkcEShuVRIjWiARKHR7kgm/MBEoe7ESGd4AESiDCJHi+AARKJejEgsfABEorfASDl6gESi+6xIcVcARKMqJEiM/4BEo5nsSBnbAESjsT
hIk6CgRKQedEgWd0BEpEo8SOeJQESkacxIeIzARKTZrEiQ0UBEpPcySAZKAESlIIpIqWXgRKU9/ko29nxEpVoYSbN56ESlcE5IgqPARKWE2EeZ54BEpaLwR5nTAESlyiBH/8QARKZPbEkpeZBEpnokSDC3wESmxRRHJcMARKcpREiYj4BEp2fmR/Cl
AESne4pI0BygRKe7eEjDdkBEp+I4SHIiwESoImploaoARKhgTEgbngBEqH+sSWpWwESonURI9cbgRKjDtEikMmBEqONAR1+pAESpKEZH7ZyARKlbTEe3BIBEqZwISYoJSESptshB4AAARKnRBEjmKeBEqfb8SM65gESqNrRICExARKqwpkgXfQBE
qswqR961gESrf4BI4hcgRKvOxEkO1VBEq+YkR0NVAESsDrxI3N9ARKxnUkiAsABErMn8SP1PwESs+ERIL3HARK0eVkh2wkBErTwgSGfcwESta65JNybQRK1/EkKKAABErfeGSPjMYESuOgBJTrywRK5Y0khxq8BErm8gQgQAAESupaZIgbJARK7H/
EfGCgBEry1cSJ5HQESVbFhlhL1ARK+AyEhNSIBEr83+SNFFwESwLtRIg6dgRLBqlEi/YkBEsIsYSII0YESw1yZJZKwQRLDszECgAABEsRXYSHB4wESxLDZDEgAARLF8eEkEA5BEsaCgSE75QESxv8hKNnd8RLHhjElfYUBEsf5gSQ3h8ESyQ7pIL4yAR
LJgbkgumEBEsqQSSRzzQESyx4ZH8B4ARLL2xkkmTRBEszSGSMPh4ESzSzBHuNuARLNsiEiJNcBEs5HASD7cgESzqY5IllUgRLP/AEeBHYBEtGz+SDrCAES0qnZI/W7gRLT1hEh7ykBEtQuoQXAAAES1NcJIiFSARLVQYEiU2aBEtY2ySMQb4ES110
5IteqARLYLSkiF8SBEtkc2SNYYoES3OG5I4zKARLefxkgtY4BEuAQAR8xzgES46o5JF2bQRLkK7EjJxGBEudC+SCaAwES6LyBIk5DgRLpXjkfQmIBEutRSSOG1QES7EkxIoskA

Base64

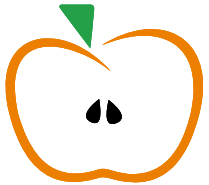
<table>
<tr><td rowspan="2">D</td><td>Text (ASCII)</td><td colspan="8">M</td><td colspan="8">a</td><td colspan="8" rowspan="2"></td></tr>
<tr><td>Octets</td><td colspan="8">77 (0x4d)</td><td colspan="8">97 (0x61)</td></tr>
<tr><td colspan="2">Bits</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="3">E</td><td>Sextets</td><td colspan="6">19</td><td colspan="6">22</td><td colspan="6">4</td><td colspan="6">Padding</td></tr>
<tr><td>Character</td><td colspan="6">T</td><td colspan="6">W</td><td colspan="6">E</td><td colspan="6">=</td></tr>
<tr><td>Octets</td><td colspan="6">84 (0x54)</td><td colspan="6">87 (0x57)</td><td colspan="6">69 (0x45)</td><td colspan="6">61 (0x3D)</td></tr>
</table>

Adapted from https://en.wikipedia.org/wiki/Base64

Solution

```
[0]  Base64Decode←{
[1]  ⍝ ⍵ ↔ Base64 encoded text
[2]  ⍝ ← ↔ Original text
[3]      t←⎕A,⎕C ⎕A,⎕D,'+/'
[4]      b←,⍉(6⍴2)⊤t⍳⍵
[5]      p←{⍵⊂⍨(≠⍵)⍴8↑1}b
[6]      XPadding←⍵∘{⍵↓⍨-+/∧\⌽'='=⍺}
[7]      ⎕UCS 2⊥¨XPadding p
[8]  }
```

```
[0]  Base64Encode←{
[1]  ⍝ ⍵ ↔ Text to encode
[2]  ⍝ ← ↔ Base64 encoded text
[3]      b←,⍉(8⍴2)⊤⎕UCS ⍵
[4]      PadDelta←{⍵-⍨⍺×⌈⍵÷⍺}
[5]      Pad←{d x←⍺ ⋄ ⍵,(d PadDelta≠⍵)⍴x}
[6]      p←{⍵⊂⍨(≠⍵)⍴6↑1}6 0 Pad b
[7]      t←⎕A,⎕C ⎕A,⎕D,'+/'
[8]      4 '='Pad t[2⊥¨p]
[9]  }
```



Tzu-Ching Lee Solution

```
_U_Fmt_ ← {
    q r ← ⌈ωω(÷⍨,|∘-)≠d ← ω∩α
    (-r)↓∈⍉ αα ⍉(q ωω)⍴d,r⍴¯1↑α
}
```

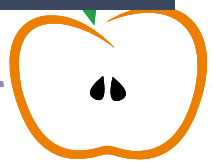
```
⍝ ┌──────────┐
⍝ | Pad &    |
⍝ | filter   |
⍝ └──────────┘
```

```
_CharEncByte_ ← {
    a ← 1⌽⍳256
    Tsf ← (ωω⍴≠α)⍤(αα⍴≠α)⊥α∘⍳
    Dec ← (Tsf⍣ 1)_U_Fmt_ αα
    Enc ← (Tsf⍣¯1)_U_Fmt_ ωω
    2|⎕DR ω:α Enc ω ⋄ α Dec ω
}
```

```
Base64 ← {
    c ← ⎕A,(⎕C⎕A),⎕D,'+/' ⍝ A-Za-z0-9+/
    r ← c (4 _CharEncByte_ 3) ω
    2|⎕DR r : r ⋄ r,(4|-≠r)⍴'='
}

Base85 ← (5 _CharEncByte_ 4)
```

Java

```java
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;
import java.util.Arrays;

public class Base64 {

    private static final char[] alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/".toCharArray();

    static String base64(InputStream is) throws IOException {
        StringBuilder sb = new StringBuilder();
        int blocks = 0;

        while (true) {
            int c0 = is.read();
            if (c0 == -1)
                break;
            int c1 = is.read();
            int c2 = is.read();

            int block = ((c0 & 0xFF) << 16) | ((Math.max(c1, 0) & 0xFF) << 8) | (Math.max(c2, 0) & 0xFF);

            sb.append(alpha[block >> 18 & 63]);
            sb.append(alpha[block >> 12 & 63]);
            sb.append(c1 == -1 ? '=' : alpha[block >> 6 & 63]);
            sb.append(c2 == -1 ? '=' : alpha[block & 63]);

            if (++blocks == 19) {
                blocks = 0;
                sb.append('\n');
            }
        }

        if (blocks > 0)
            sb.append('\n');

        return sb.toString();
    }
```



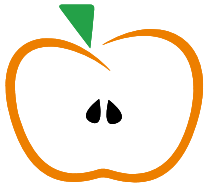
VBA

```
Option Explicit
Public Function Decode(s As String) As String
    Dim i As Integer, j As Integer, r As Byte
    Dim FirstByte As Byte, SecndByte As Byte, ThirdByte As Byte
    Dim SixBitArray() As Byte, ResultString As String, Token As String
    Dim Counter As Integer, InputLength As Integer
    InputLength = Len(s)
    ReDim SixBitArray(InputLength + 1)
    j = 1  'j counts the tokens excluding cr, lf and padding
     For i = 1 To InputLength 'loop over s and translate tokens to 0-63
        Token = Mid(s, i, 1)
        Select Case Token
            Case "A" To "Z"
                SixBitArray(j) = Asc(Token) - Asc("A")
                j = j + 1
            Case "a" To "z"
                SixBitArray(j) = Asc(Token) - Asc("a") + 26
                j = j + 1
            Case "0" To "9"
                SixBitArray(j) = Asc(Token) - Asc("0") + 52
                j = j + 1
            Case "+"
                SixBitArray(j) = 62
                j = j + 1
            Case "/"
                SixBitArray(j) = 63
                j = j + 1
            Case "="
                'padding'
            Case Else
                'cr and lf
        End Select
    Next i
    r = (j - 1) Mod 4
    Counter = 1
    For i = 1 To (j - 1) \ 4 'loop over the six bit byte quadruplets
        FirstByte = SixBitArray(Counter) * 4 + SixBitArray(Counter + 1) \ 16
        SecndByte = (SixBitArray(Counter + 1) Mod 16) * 16 + SixBitArray(Counter + 2) \ 4
        ThirdByte = (SixBitArray(Counter + 2) Mod 4) * 64 + SixBitArray(Counter + 3)
        ResultString = ResultString & Chr(FirstByte) & Chr(SecndByte) & Chr(ThirdByte)
        Counter = Counter + 4
    Next i
    Select Case r
        Case 3
            FirstByte = SixBitArray(Counter) * 4 + SixBitArray(Counter + 1) \ 16
            SecndByte = (SixBitArray(Counter + 1) Mod 16) * 16 + SixBitArray(Counter + 2) \ 4
            ResultString = ResultString & Chr(FirstByte) & Chr(SecndByte)
        Case 2
            FirstByte = SixBitArray(Counter) * 4 + SixBitArray(Counter + 1) \ 16
            ResultString = ResultString & Chr(FirstByte)
    End Select
    Decode = ResultString
End Function
```

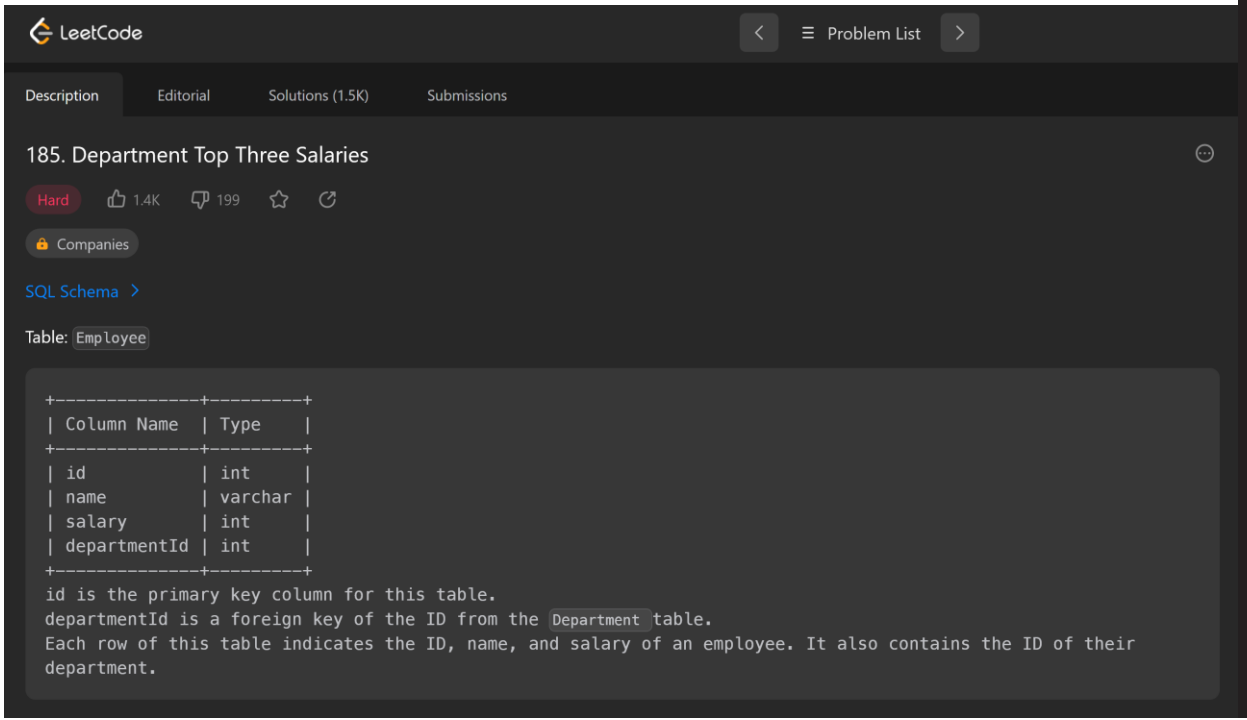
```
Public Function Encode(s As String) As String
    Dim InputLength As Integer, FirstByte As Byte, SecndByte As Byte, ThirdByte As Byte, r As Byte
    Dim LineNumber As Integer, z As Integer, q() As String, ResultString As String
    Dim FullLines As Integer, LastLineLength As Integer, Counter As Integer
    q = Split("A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,a,b,c,d,e,f,g,h,i,j," & _
        "k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,0,1,2,3,4,5,6,7,8,9,+,/", ",", -1, vbTextCompare)
    InputLength = Len(s)
    r = InputLength Mod 3
    FullLines = ((InputLength - r) / 3) \ 20 + 1: LastLineLength = (InputLength - r) / 3 Mod 20 - 1
    Counter = 1
    For LineNumber = 1 To FullLines
        For z = 0 To IIf(LineNumber < FullLines, 19, LastLineLength) 'loop over the byte triplets
            FirstByte = Asc(Mid(s, Counter, 1))
            SecndByte = Asc(Mid(s, Counter + 1, 1))
            ThirdByte = Asc(Mid(s, Counter + 2, 1))
            Counter = Counter + 3
            ResultString = ResultString & q(FirstByte \ 4) & q((FirstByte Mod 4) * 16 + _
                (SecndByte \ 16)) & q((SecndByte Mod 16) * 4 + (ThirdByte \ 64)) & q(ThirdByte Mod
64)
        Next z
        If LineNumber < FullLines Then ResultString = ResultString & vbCrLf
    Next LineNumber
    Select Case r
        Case 2
            FirstByte = Asc(Mid(s, Counter, 1))
            SecndByte = Asc(Mid(s, Counter + 1, 1))
            ResultString = ResultString & q(FirstByte \ 4) & q((FirstByte Mod 4) * 16 + _
                (SecndByte \ 16)) & q((SecndByte Mod 16) * 4) & "="
        Case 1
            FirstByte = Asc(Mid(s, Counter, 1))
            ResultString = ResultString & q(FirstByte \ 4) & q((FirstByte Mod 4) * 16) & "=="
    End Select
    Encode = ResultString
End Function
```

JavaScript

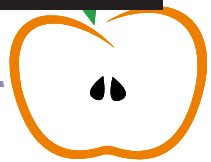
```
[1] (function(){//ECMAScript doesn't have an internal base64 function or method, so we have to do it ourselves, isn't that exciting?
[2]     function stringToArrayUnicode(str){for(var i=0,l=str.length,n=[];i<l;i++)n.push(str.charCodeAt(i));return n;}
[3]     function generateOnesByLength(n){//Attempts to generate a binary number full of ones given a length.. they don't redefine each other that much.
[4]         var x=0;
[5]         for(var i=0;i<n;i++){
[6]             x<<=1;x|=1;//I don't know if this is performant faster than Math.pow but seriously I don't think I'll need Math.pow, do I?
[7]         }
[8]         return x;
[9]     }
[10]     function paf(_offset,_offsetlength,_number){//I don't have any name for this function at ALL, but I will explain what it does, it takes an offset, a number and returns the base64 number and the offset of the next number.
[11]         //the next function will be used to extract the offset of the number..
[12]         var a=6-_offsetlength,b=8-a;//Oh god, 8 is HARDCODED! Because 8 is the number of bits in a byte!!!
[13]         //And 6 is the mini-byte used by wikipedia base64 article... at least on 2013.
[14]         //I imagine this code being read in 2432 or something, that probably won't happen..
[15]         return [_number&generateOnesByLength(b),b,(_offset<<a)|(_number>>b)];//offset & offsetlength & number
[16]     }
[17]     function toBase64(uint8array){//of bits, each value may not have more than 255 bits... //a normal "array" should work fine too..
[18]         //From 0x29 to 0x5a plus from 0x61 to 0x7A AND from 0x30 to 0x39
[19]         //Will not report errors if an array index has a value bigger than 255.. it will likely fail.
[20]         var a=[],i,output=[];
[21]         for(i=0x41;i<=0x5a;i++){//A-Z
[22]             a.push(String.fromCharCode(i));
[23]         }
[24]         for(i=0x61;i<=0x7A;i++){//a-z
[25]             a.push(String.fromCharCode(i));
[26]         }
[27]         for(i=0x30;i<=0x39;i++){//0-9
[28]             a.push(String.fromCharCode(i));
[29]         }
[30]         a.push('+','/');
[31]         var offset=0,offsetLength=0,x;
[32]         for(var i=0,l=uint8array.length;i<l;i++){
[33]             if(offsetLength==6){//if offsetlength is 6 that means that a whole offset is occupying the space of a byte, can you believe it.
[34]                 offsetLength=0;
[35]                 output.push(a[offset]);
[36]                 offset=0;
[37]                 i--;
[38]                 continue;
[39]             }
[40]             x=paf(offset,offsetLength,uint8array[i]);
[41]             offset=x[0];
[42]             offsetLength=x[1];
[43]             output.push(a[x[2]]);
[44]         }
[46]             if(offsetLength==6){
[47]                 output.push(a[offset]);
[48]             }else{
[49]                 var y=(6-offsetLength)/2;
[50]                 x=paf(offset,offsetLength
,0);
[51]                 offset=x[0];
[52]                 output.push(a[x[2]]);
[53]                 switch (y){
[54]                     case
2:output.push('=');//This thingy right here, you
know.. the offsets also, no break statement;
[55]                     case
1:output.push('=');break;
[56]                 }
[57]             }
[58]         }
[59]         return output.join('');//You can
change it so the result is an array instead!!!!
[60]     }
```



LeetCode 185 – SQL




```
Input:
Employee table:
+----+-------+--------+--------------+
| id | name  | salary | departmentId |
+----+-------+--------+--------------+
| 1  | Joe   | 85000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
| 5  | Janet | 69000  | 1            |
| 6  | Randy | 85000  | 1            |
| 7  | Will  | 70000  | 1            |
+----+-------+--------+--------------+
Department table:
+----+-------+
| id | name  |
+----+-------+
| 1  | IT    |
| 2  | Sales |
+----+-------+
Output:
+------------+----------+--------+
| Department | Employee | Salary |
+------------+----------+--------+
| IT         | Max      | 90000  |
| IT         | Joe      | 85000  |
| IT         | Randy    | 85000  |
| IT         | Will     | 70000  |
| Sales      | Henry    | 80000  |
| Sales      | Sam      | 60000  |
+------------+----------+--------+
Explanation:
In the IT department:
- Max earns the highest unique salary
- Both Randy and Joe earn the second-highest unique salary
- Will earns the third-highest unique salary

In the Sales department:
- Henry earns the highest salary
- Sam earns the second-highest salary
- There is no third-highest salary as there are only two employees
```

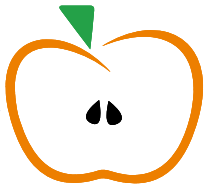


SQL

```
SELECT e1.Name, e1.Salary
FROM emp e1
WHERE 3 >
  (SELECT COUNT(DISTINCT e2.Salary)
      FROM emp e2
      WHERE e2.Salary > e1.Salary
       AND e1.dept = e2.dept
  );
```

FlipDB

3 > 1 rankDown by Salary (group DepartmentID)



LeetCode 571: Find Median Given Frequency of Numbers

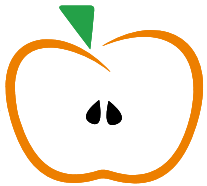
Num	Frequency
0	7
1	1
2	3
3	1

FlipDB

median Frequency replicate Num

SQL

```
select avg(Number) as median from (
    select n1.Number from Numbers n1 join Numbers n2 on n1.Number >= n2.Number
        group by n1.Number
        having
        sum(n2.Frequency) >= (select sum(Frequency) from Numbers) / 2
        and
        sum(n2.Frequency) - avg(n1.Frequency) <= (select sum(Frequency) from Numbers) / 2
) med;
```



References

- https://www.kaggle.com/datasets/adityakadiwal/water-potability
- https://en.wikipedia.org/wiki/Base64
- https://rosettacode.org/wiki/Base64_encode_data
- https://www.dyalog.com/uploads/conference/dyalog22/presentations/U15_How_I_%20Won_the_APL_Problem_Solving_Competition.pdf
- https://www.toolofthought.com/
- https://leetcode.com/problems/department-top-three-salaries