CTI *2016 Nov 18*

Hint : in order to solve these problems, always think about the inputs and outputs, and then the recursion.

About Algorithm

Question 1 : Write a function to remove space from a string inplace. *[Similar to **Volant Trading** question 1]*

```
void remove_space(std::string& str)
{
      unsigned short m=0;
      for(unsigned short n=0; n!=str.size(); ++n)
      {
            if (str[n]!=' ')
            {
                  if (m!=n) str[m] = str[n];
                  ++m;
            }
      }
      str.erase(str.begin()+m, str.end()); // This implementation is tested in MSVS.
}
```

Question 2 : Write a function to return the Nth row of Pascal triangle (for example, the 5th row is 1-4-6-4-1).

```
std::vector<int> pascal_vector(int n)
{
      if (n==0) return std::vector<int>();
      if (n==1) return std::vector<int>(1,1);

      std::vector<int> row;
      auto v = pascal_vector(n-1);

      row.push_back(1);
      for(int n=0; n!=v.size()-1; ++n) row.push_back(v[n]+v[n+1]);
      row.push_back(1);
      return row;
}
```

Question 3 : Given a vector of orders sorted by order IDs, write a function to build a balanced binary tree of orders.

```
template<typename ITER>
node<typename std::iterator_traits<ITER>::value_type>* build_tree(ITER begin, ITER end)
{
      typedef typename std::iterator_traits<ITER>::value_type T;
      if (begin == end) return nullptr;

      ITER mid = begin + (end-begin)/2;
      node<T>* output = new node<T>(*mid);
      output->lhs = build_tree(begin, mid);
      output->rhs = build_tree(mid+1, end);
      return output;
}
```

Question 4 : Given a set of stock codes and last price, write an algorithm to return stock code with Nth highest last price. Try not to use other libraries.

```
template<unsigned short N> void topN<N>(ITER begin, ITER end)
{
      for(ITER i=begin; i!=begin+N; ++i)
      {
            for(ITER j=i+1; j!=end; ++j)
            {
                  if (*i<*j) std::swap(*i,*j);
            }
      }
}
```

About C++

Question 1 : Design your own shared pointer.

```
template<typename T>
class shared_ptr
{
public:
      shared_ptr() : ptr(nullptr), ref_count_ptr(nullptr) {}

      // malloc once by caller for _ptr
      // malloc once by shared_ptr for manager
      explicit shared_ptr(T* _ptr) : ptr(_ptr), ref_count_ptr(new int(1)) {}
      explicit shared_ptr(shared_ptr<T> rhs) : ptr(nullptr), ref_count_ptr(nullptr)
      {
            increment(rhs);
      }

      shared_ptr<T>& operator=(shared_ptr<T> rhs)
      {
            decrement();
            increment(rhs);
            return *this;
      }

      ~shared_ptr()
      {
            decrement();
      }

      T& operator*()     { return*ptr; }
      T* operator->()    { return ptr; }

private:
      void increment(shared_ptr<T> rhs)
      {
            ptr = rhs.ptr;
            ref_count_ptr = rhs.ref_count_ptr;
            if (ref_count_ptr!=nullptr) ++(*ref_count_ptr);
      }

      void decrement()
      {
            if (ref_count_ptr!=nullptr && *ref_count_ptr>0)
            {
                  -- (*ref_count_ptr);
                  if (*ref_count_ptr==0)
                  {
                        delete ptr;               ptr = nullptr;
                        delete ref_count_ptr;     ref_count_ptr = nullptr;
                  }
            }
      }

private:
      T* ptr;
      unsigned long* ref_count_ptr; // also known as manager
};
// Please provide make_shared which perform malloc once instead of twice.
```

Question 2 : What are the printout?

```
class B0 { public: virtual void connect()   { std::cout << "\nB0::connect";}
                   virtual void logon()     { std::cout << "\nB0::logon";   }};
class B1 { public: virtual void connect()   { std::cout << "\nB1::connect";}
                   virtual void logon()     { std::cout << "\nB1::logon";   }};
class B2 { public: virtual void connect()   { std::cout << "\nB2::connect";}
                   virtual void logon()     { std::cout << "\nB2::logon";   }};

class D :  public B0, public B1, public B2
{
           public: virtual void connect()   { std::cout << "\nD::connect"; }
};

B0 b0;                   B0* pb0 = &b0;
B1 b1;                   B1* pb1 = &b1;
B2 b2;             //    B2* pb2 = &b1;     // compile error
D  d;

b0.connect();            b0.logon();
b1.connect();            b1.logon();
d.connect();       //    d.logon();         // ambiguous
pb0->connect();          pb0->logon();
pb1->connect();          pb1->logon();
```

Question 3 : What are the printout?

```
      class A
      {     public:
            A()               { std::cout << "\ndefault constructor";     }
            A(const A&)       { std::cout << "\ncopy constructor";        }
            A(int)            { std::cout << "\nconversion constructor";  }
      };

      A a0;                   // print default constructor
      A a1(a0);               // print copy constructor           note : direct initialization
      A a2 = a0;              // print copy constructor           note : copy initialization
      A a3(123);              // print conversion constructor     note : direct initialization
      A a4 = 123;             // print conversion constructor     note : copy initialization (creat temp obj from int)
```

If class A is slightly modified as the following, what are the printout?

```
      class A
      {     public:
            A()               { std::cout << "\ndefault constructor";     }
            A(A&)             { std::cout << "\ncopy constructor";        }
            A(int)            { std::cout << "\nconversion constructor";  }
      };

      A a0;                   // print default constructor
      A a1(a0);               // print copy constructor           note : direct initialization
      A a2 = a0;              // print copy constructor           note : copy initialization
      A a3(123);              // print conversion constructor     note : direct initialization
//    A a4 = 123;             // compile error
```

About Linux

Question 1 : Setup a cron job to send email to xxx@cashalgo.com the number of orders in ~/tmp/neworder.csv once every hour. Assuming that order IDs are unique. Here is an example of csv file. Note : csv = comma separated values.

```
      timestamp,exchange,order_id,contract,price,status
      20160601093000.123,SEHK,oid001,0005,40.00,NEW
      20160601093106.409,SEHK,oid002,0005,40.10,PARTIAL_FILL
      20160601093131.081,SEHK,oid003,0005,40.10,NEW
      20160601093156.056,SEHK,oid004,0005,40.00,NEW
      20160601093210.743,SEHK,oid005,0005,40.10,FILL
      20160601093213.863,SEHK,oid006,0005,40.00,CANCELLED
```

Without crontab

```
>>    cut -f 3 -d ‘,’ ~/tmp/neworder.csv | uniq | wc -l | mail -s “Subject: #orders” xxx#cashalgo.com
```

With crontab

```
>>    0 * * * * cut -f 3 -d ‘,’ ~/tmp/neworder.csv | uniq | wc -l | mail -s “Subject: #orders” xxx#cashalgo.com
```

Question 2 : On a Centos server with 2GB RAM, sort a tick data csv file with size 6GB by the second column.

```
>>    split -b 2GB data.csv splited_
>>    cut -f 1-  -d ‘,’ --output-delimiter=‘ ’ splited_aa | sort -k 2 > splited_and_sorted_aa
>>    cut -f 1-  -d ‘,’ --output-delimiter=‘ ’ splited_ab | sort -k 2 > splited_and_sorted_ab
>>    cut -f 1-  -d ‘,’ --output-delimiter=‘ ’ splited_ac | sort -k 2 > splited_and_sorted_ac
>>    sort -m splited_and_sorted_* > sorted.csv
//    (1) cut is for conversion of delimiter, as sort’s default delimiter is space.
//    (2) The final step is a merging of sorted results, no real sorting is done in this step.
```

Question 3 : Sum all integers (one per line) in a file /tmp/array.csv.

```
>>    awk ‘{ count += $1; } END { print count; }’ /tmp/array.csv
```

The above works if each line contains purely an integer. However, if the integer is mixed with other characters in a string, then we need to use grep –o (which means output the matched part only) :

```
>>    grep -o ‘[0-9]*’ /tmp/array.csv | awk ‘{ count += $1; } END { print count; }’
```