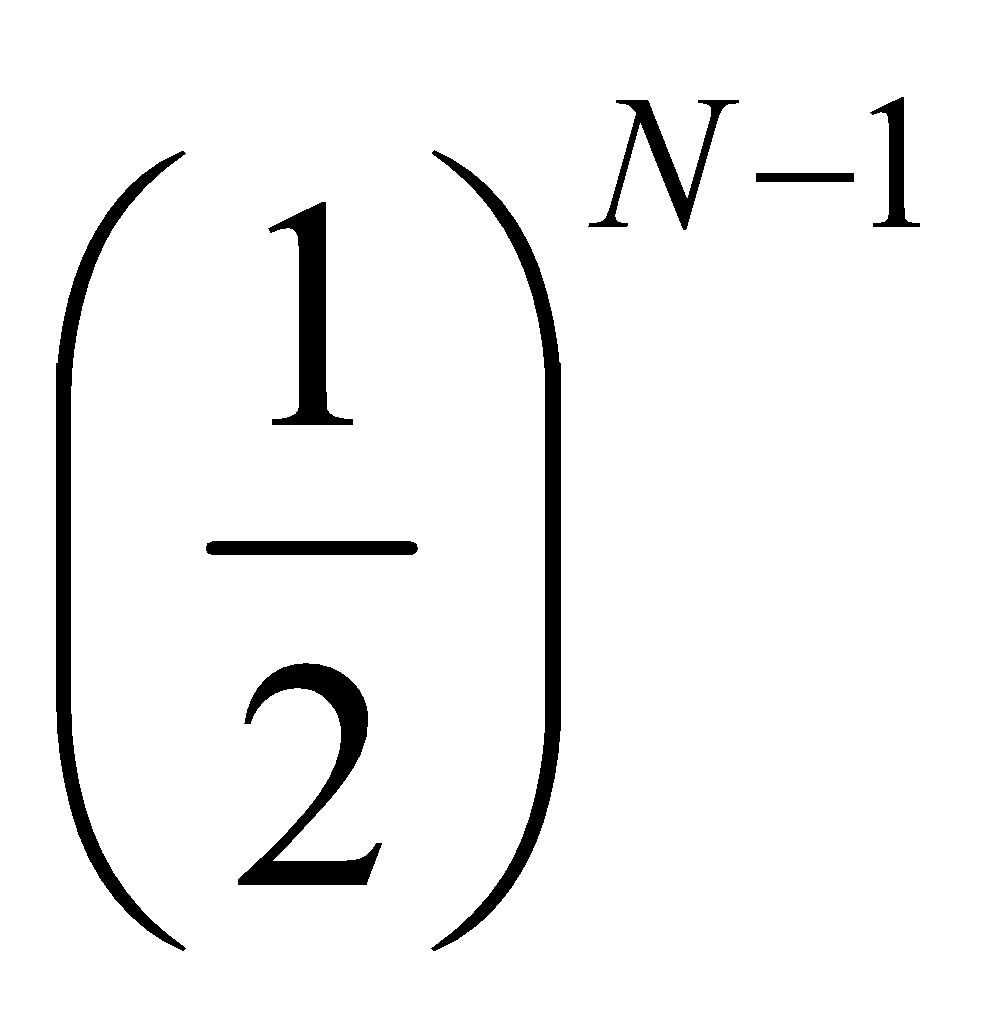
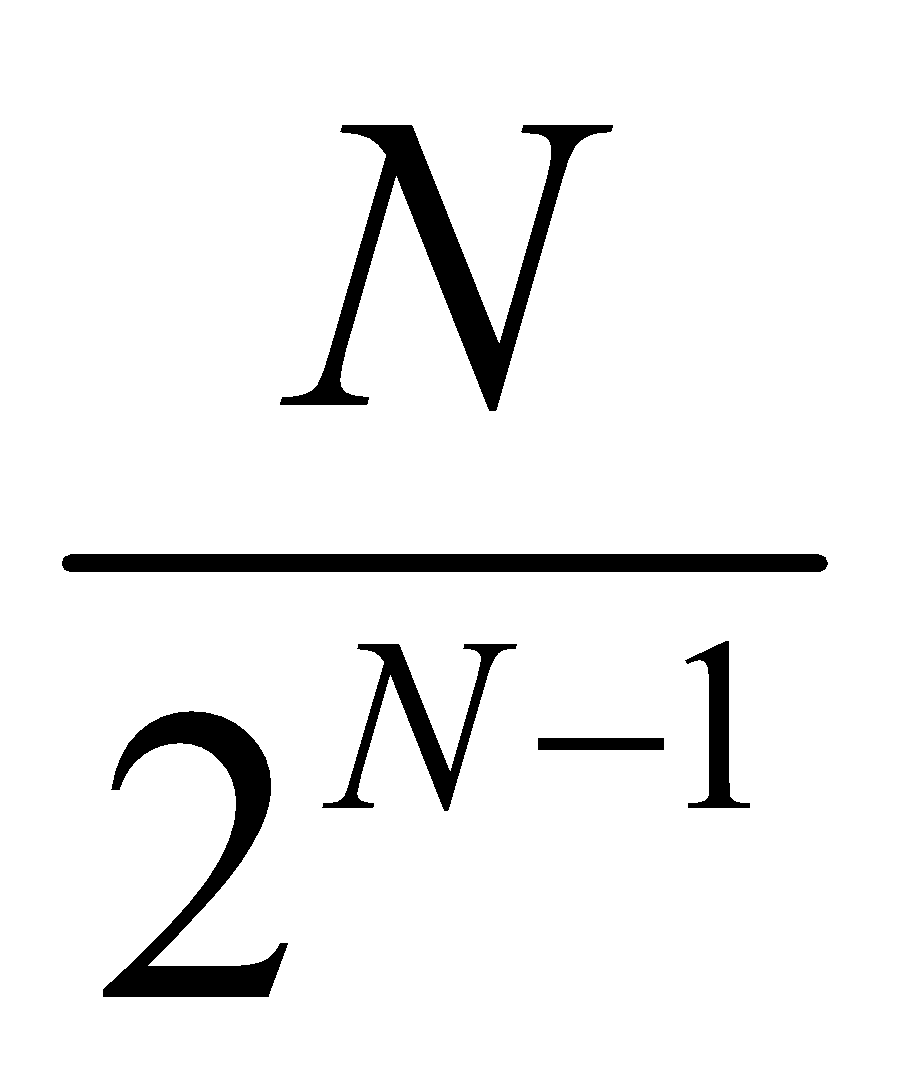
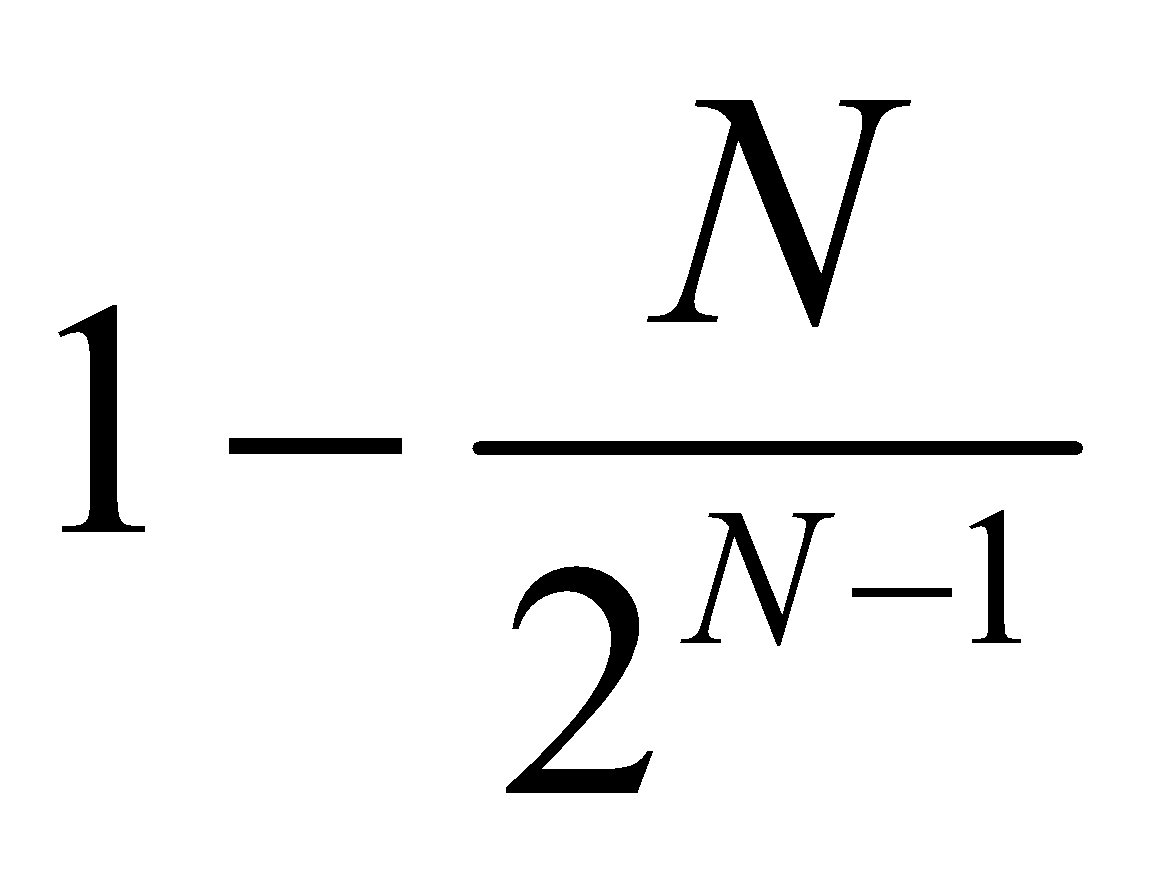
Q7.

Cutting a stick into N pieces with N-1 cut points is equivalent to cutting a same-length circle into N pieces with N cut points, since the first cut is arbitrary. To form a polygon, we need the length of any piece should be less than sum of other pieces. Assume the length of the stick and the circumference of the circle is 1, if one piece is more than 0.5, then they cannot form a polygon.

Define event Ei as from first cut at point i the other points are in the clockwise semicircle.

P(Ei) = , for i = 1...N.

P(cannot form a polygon) = 

P(can form a polygon) = 

Ref: <http://www.zhihu.com/question/25408010> and Math.pdf

Q10.

A number of power of 2 has form of 10...0 and the number minus 1 has form of 01...0. If we and them bitwise, it will return 0. We also want to exclude 0. C++ Code:

bool isPower2(int x)

{

return x && !(x & (x - 1));

}

Ref: <http://stackoverflow.com/questions/3638431/determine-if-an-int-is-a-power-of-2-or-not-in-a-single-line>

Q14.

We use the straightforward way to compare the substring with the whole string. If the first character of substring appears, it starts to count until finished the comparison. If the comparison is interrupted, we will move forward to later comparison. C++ Code:

#include <iostream>

#include <string>

#include <vector>

using namespace std;

vector<int> indexOf(string& str, string& substr)

{

vector<int> ivec; // use vector to contain match index

int len = str.size();

int sublen = substr.size();

int limit = len - sublen + 1;

for (int i = 0; i < limit; i++)

{

int count = 0;

for (int j = 0; j < sublen; j++)

{

if (str[j+i] != substr[j])

{

i += j; // increase i by j if not match

break;

}

count++;

}

if (count == sublen)

{ ivec.push\_back(i);}

}

return ivec;

}

int main()

{

string s = "qishibuysidequantshixinhang";

string p = "shi";

vector<int> ivec = indexOf(s,p); // match at 2 and 17

for (int i = 0; i < ivec.size(); i++)

{

cout << ivec[i] << ' ';

}

return 0;

}

Q15.

We use Taylor expansions to expand the exponential function. Since the expansion terms will be smaller and smaller, we stop the approximation with the term is smaller than the value we defined. C++ Code:

#include <iostream>

using namespace std;

// use Taylor expansion to approximate the e^x

double expo(double x)

{

double ret = 0;

double t = 1;

// stop approximation if the item is small enough

for (int i = 1;t > 1.0E-20;i++)

{

ret += t;

t = t \* x / i;

}

return ret;

}

int main()

{

double j = expo(20);

cout << j << endl;

return 0;

}

Q17.

If we can both long and short stocks, the long short profit (lsp) will be the sum of absolute difference of any two consecutive numbers. If we can only long stocks, the long profit (lp) will be the sum of positive difference of any two consecutive numbers. We also attach the code to find the buy and sell day numbers when only long is permitted.

Python Code to find max profit:

plist = [10,11,12,13,14,12,12,11,12,14,13,11]

lp = 0 # max profit for only long

lsp = 0 # max profit for both long and short

for i in range(len(plist)-1):

if plist[i] < plist[i+1]:

lp += plist[i+1] - plist[i];

lsp += plist[i+1] - plist[i];

else:

lsp += plist[i] - plist[i+1]

print lp,lsp

Python Code to find buy and sell point:

plist = [10,11,12,13,14,12,12,11,12,14,13,11]

blist = []

slist = []

pos = False

if plist[0] < plist[1]:

pos = True

blist.append(1)

for i in range(1,len(plist)-1):

if plist[i-1] <= plist[i] and plist[i] > plist[i+1] and pos:

slist.append(i+1)

pos = False

if plist[i-1] >= plist[i] and plist[i] < plist[i+1] and not pos:

blist.append(i+1)

pos = True

if pos:

slist.append(len(plist)+1)

print 'Buy at ',blist,'\nSell at ',slist