```cpp
int n;
double sum,a;
void work()
{
    while(cin>>a)
    {
        n++;
        sum+=a;
    }
    printf("￥%.2lf",sum/n);
    return;
}
```

```python
import base64

def cov64(s):
    while((s[0:5]!='flag{') or (s[-1]!='}')):
        s=base64.b64decode(s).decode()
    return s

print(cov64(input()))
```

```cpp
#define int long long
string s;
int n,m,ans;
const int MOD=19260817;
stack<int> st;
signed main()
{
    cin>>n>>m>>s;
    for(int i=0;i<m;i++)
    {
        if(s[i]=='(') st.push(0);
        else
        {
            int ret=st.top();
            st.pop();
            if(ret==0)
            {
                if(!st.empty()) st.top()++;
                else ans=(ans+1)%MOD;
            }
            else
            {
                if(!st.empty()) st.top()=(ret*n+st.top())%MOD;
                else ans=(ans+ret*n)%MOD;
            }

        }
    }
    cout<<ans<<endl;
    return 0;
}
```

```cpp
int l,T;
string str;
int main()
{
    ios::sync_with_stdio(false);
    cin>>T;
    while(T--)
    {
        cin>>l>>str;
        int i=0,j=1;
        str=str+str;
        while(j<l)
        {
            int k=0;
            while(k<l && str[j+k]==str[i+k]) k++;
            if(k==l) break;
            if(str[j+k]<str[i+k]) i=max(j+1,i+k+1);
            else j=j+k+1;
            if(i>j) swap(i,j);
        }
        cout<<i<<endl;
    }
    return 0;
}
```

```cpp
int n,m,sx,sy;
char g[23][23];
int dx[4]={0,0,1,-1},dy[4]={1,-1,0,0};

void DFS(int x, int y)
{
    if(x<0 || x>=n || y<0 || y>=m || g[x][y]=='*' || g[x][y]=='v') return;
    if(!x)
    {
        cout<<"So dalao, Re Bu Qi."<<endl;
        exit(0);
    }
    g[x][y]='v';
    for(int i=0;i<4;i++) DFS(x+dx[i],y+dy[i]);
}

int main()
{
    cin>>n>>m;
    for (int i=0; i<n; i++)
        for(int j=0; j<m; j++)
        {
            cin>>g[i][j];
            if(g[i][j]=='S') sx=i,sy=j;
        }
    DFS(sx,sy);
    cout<<"Man Shen Chuang Yi."<<endl;
    return 0;
}
```

```cpp
const int N = 1e4+7;

template <typename T>
struct SegmentTree {
  int sz;
  T tr[N<<2], lazy[N<<2];
  SegmentTree(){}
  void build(const int &n, const T &k = 0) { sz = n; _build(1, n, k); }
  template <typename TT>
  void build(const TT a[], const int &n) { sz = n; _build(a, 1, n); }
  void modify(const int &x, const T &k) { _modify(x, k, 1, sz); }
  void add(const int &x, const T &k) { _add(x, x, k, 1, sz); }
  void add(int l, int r, const T &k) { if (l > r) swap(l, r); _add(l, r, k, 1, sz); }
  T query(const int &x) { return _query(x, x, 1, sz); }
  T query(int l, int r) { if (l > r) swap(l, r); return _query(l, r, 1, sz); }
private :
  void push_up(const int &i) { tr[i] = tr[i<<1]+tr[i<<1|1]; }
  void push_down(const int &i, const int &len) {
    if (!lazy[i]) return;
    tr[i<<1] += lazy[i]*(len-len/2);
    tr[i<<1|1] += lazy[i]*(len/2);
    lazy[i<<1] += lazy[i];
    lazy[i<<1|1] += lazy[i];
    lazy[i] = 0;
  }
  void _build(const int &l, const int &r, const T &k = 0, const int &i = 1) {
    lazy[i] = 0;
    if (l == r) { tr[i] = k; return; }
    int mid = (l+r)>>1;
    _build(l, mid, k, i<<1);
    _build(mid+1, r, k, i<<1|1);
    push_up(i);
  }
  template <typename TT>
  void _build(const TT a[], const int &l, const int &r, const int &i = 1) {
    lazy[i] = 0;
    if (l == r) { tr[i] = a[l]; return; }
    int mid = (l+r)>>1;
    _build(a, l, mid, i<<1);
    _build(a, mid+1, r, i<<1|1);
    push_up(i);
  }
  void _modify(const int &x, const T &k, const int &trl, const int &trr, const int &i = 1) {
    if (trl == x && trr == x) {
      tr[i] = k;
      lazy[i] = 0;
      return;
    }
    push_down(i, trr-trl+1);
    int mid = (trl+trr)>>1;
    if (x <= mid) _modify(x, k, trl, mid, i<<1);
    else _modify(x, k, mid+1, trr, i<<1|1);
    push_up(i);
```

```
  }
  void _add(const int &l, const int &r, const T &k, const int &trl, const int &trr, const int &i = 1) {
    if (trl >= l && trr <= r) {
      tr[i] += k*(trr-trl+1);
      lazy[i] += k;
      return;
    }
    push_down(i, trr-trl+1);
    int mid = (trl+trr)>>1;
    if (l <= mid) _add(l, r, k, trl, mid, i<<1);
    if (r >  mid) _add(l, r, k, mid+1, trr, i<<1|1);
    push_up(i);
  }
  T _query(const int &l, const int &r, const int &trl, const int &trr, const int &i = 1) {
    if (trl >= l && trr <= r) return tr[i];
    push_down(i, trr-trl+1);
    int mid = (trl+trr)>>1;
    T res = 0;
    if (l <= mid) res += _query(l, r, trl, mid, i<<1);
    if (r >  mid) res += _query(l, r, mid+1, trr, i<<1|1);
    return res;
  }
};

int n;
int a[N];
SegmentTree<int> tree;

signed main() {
  ios::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
  while (scanf("%d", &n) == 1) {
    tree.build(1e4);
    for (int i = 1; i <= n; ++i) {
      scanf("%d", a+i);
      tree.add(a[i], 1);
    }
    double s = 0;
    for (int i = 1, mx = 2; i <= n; ++i) {
      tree.add(a[i], -1);
      if (a[i] < mx) continue;
      s += tree.query(1, a[i]-1);
      mx = max(mx, a[i]);
    }
    printf("%.5f\n", s/(s+n));
  }
  return 0;
}
```