1. **Maximum Positivity**:

   Topic: Maths, Two pointer
   We can use this problem using a single while loop:-

   For all elements in array :-

   1.If ith element is negative, we need to ignore it and go on next element

   2. If ith element is non-negative, we will start a second while loop from this position until a negative element arrives.
      a.If size of subarray received using this is greater than size of previous such arrays, then update the answer
      b. else ignore it.

2. **Area under the hills**
   Topic: Maths

   This problem can be solved using 2 ways.

I. for every next subsequent point there always are 2 regions: 1 rectangle and 1 triangle. You can see that every time min would be added as a rectangle and maximum-minimum would be added as triangle. Now base always remains 1. So you always add minimum + (maximum-minimum)/2 or (max+min)/2 for each consecutive point.

II. You can use your observation to note that answer is always sum of numbers. Which can be proved from above argument.
Complete Solution:

```
string solve(vector<int> A)
{
    long long ans=0;
    for(int x:A)
        ans+=(long long) x;
    return to_string(ans);
}
```

3. **Maximum Product**

Topic:- Dp, array

Observation 1:- Maximum sum of all array would be always <= 100000.

Observation 2:- Now, if we have all possible sums , our answer would be maximum of ((sums) * (total sum of array - sums)) where sum iterates over all possible sums.

To calculate all possible sums :-
To check whether a sum can be obtained by taking a subset of an array is nothing but subset sum problem and can be solved by dynamic programming , idea is :-

So it can be solved by taking a 10^5 boolean array dp. Here 10^5 is used for the sum of the elements of the array.

where dp[i] denotes whether sum i is possible or not. Initialize dp[i] for all i to be false/0. Base Case: dp[0] = 1. Recurrence is : dp[i] = dp[i-A[j]], where j goes from 1 to N and i goes from 10^5 to A[j]

4. **Mysterious Function**
   Topic: Number Theory(Euler Theorem)

   Solution Approach:

   If you were not able to solve this problem and did not know euler theorem, I would request you to first learn it and try again.
   We know,
   $A^{\varphi(m)} \equiv 1$ (modulo m)
   Implies we can write:
   $A^{X} \equiv A^{X(\text{modulo } \varphi(m))}$ (modulo m)

   Now, Question ask us to calculate $A[1]^{A[2]\ldots}$ **(modulo m)**
   $A[1]^{A[2]\ldots}$ (modulo m) $\equiv$ $A[1]^{A[2]\ldots(\text{modulo } \varphi(m))}$

   So problem reduces to calculate $A[2]^{A[3]\ldots}$ **(modulo φ(m))**
   You can solve this recursively until there exist only 1 element.

   Complete Solution:

```
int phi[40004];
void precompute()
{
    if(phi[5]==4 and phi[3]==2 and phi[2]==1)
        return;
    phi[0]=phi[1]=0;
    for(int i=2;i<=40000;i++)
        phi[i]=i;
    for(int i=2;i<=40000;i++)
        if(phi[i]==i)
            for(int j=i;j<=40000;j+=i)
            {
```

```
                phi[j]/=i;
                phi[j]*=(i-1);
            }
}
int fast_power(int x, int y, int p)
{
    int res = 1; x %= p;
    while (y > 0)
    {
        if (y & 1)
            res = (res*x) % p;
        y>>=1;
        x = (x*x) % p;
    }
    return res;
}
int recursion(vector<int> &A,int m,int idx)
{
    if(m<=1)
        return 0;
    if(idx==0)
        return A[idx]%m;
    int x=recursion(A,phi[m],idx-1);
    return fast_power(A[idx],x,m);
}
int Solution::mysterious_function(vector<int> &A, int B)
{
    precompute();
    return recursion(A,B,A.size()-1);
}
```