This is code for an algorithm which finds a maximum length strictly increasing subsequence of a sequence A[0] ... A[n-1] of integers. There could be more than one such subsequence, but the algorithm chooses just one of them.

In order to explain the algorithm, we define Length(i) to be the length of the longest strictly monotone subsequence of A[0] ... A[i]. The function Length does not appear in the program, but it is useful for explanation purposes. The arrays index and value below change during the execution of the program, but after iteration t of the for loop of the function mainwork(), index[ell] = i and value[ell] = A[i] where i is the maximum number no larger than t such that Length(i) = ell, if such an i exists. If not, index[ell] and value[ell] are undefined.

I know that's pretty hard to grasp, but it works!

```
// program to find the longest strictly monontone increaseing subsequence

int const N = 20;
int n;
int A[N]; // the input sequence
int back[N]; // back[i] = backpointer of longest incs ending at a[i]
int L; // length of longest maximal monotone increasing subsequence so for
int index[N]; // Mystery! (See the definition above.)
int value[N]; // Mystery! (See the definition above.)

void getA()
 {
  cin >> n;
  for(int i = 0; i < n; i++) cin >> A[i];
 }

void startup()
 {
  L = 1;
  index[0] = -1; // this is a fictitious value
  back[0] = -1; // this is a fictitious value
  index[1] = 0;
  value[1] = A[0];
 }

void writebackwards(int indx)
 {
  if(indx >= 0)
   {
    writebackwards(back[indx]);
    cout << " " << A[indx];
   }
 }
```

```
void writeanswer()
 {
  writebackwards(index[L]);
  cout << endl;
 }

void mainwork()
 {
  getA();
  for(int t = 0; t < n; t++)
   cout << " " << A[t]; // print the input sequence
  cout << endl;
  startup();
  for(int t = 1; t < n; t++)
   {
    int s = 1;
    while(s <= L and A[t] > value[s])s++;
    value[s] = A[t];
    index[s] = t;
    back[t] = index[s-1];
    if (s > L) L = s;
   }
  writeanswer(); // print the output sequence
 }

int main()
 {
  getA();
  mainwork();
  return 1;
 }
```

Here are four runs of the program:

```
0 5 9 3 8 7 2 5 9 6 7 3
0 2 5 6 7

1 1 2 2 3 3
1 2 3

0 6 9 8 1 3 2 5 4 7
0 1 2 4 7

1 3 5 7 9 2 4
1 3 5 7 9
```