AP Computer Science A
2000 Student Samples

The materials included in these files are intended for non-commercial use by AP teachers for course and exam preparation; permission for any other use must be sought from the Advanced Placement Program. Teachers may reproduce them, in whole or in part, in limited quantities, for face-to-face teaching purposes but may not mass distribute the materials, electronically or otherwise. These materials and any copies made of them may not be resold, and the copyright notices must be retained as they appear here. This permission does not apply to any third-party copyrights contained herein.

These materials were produced by Educational Testing Service (ETS), which develops and administers the examinations of the Advanced Placement Program for the College Board. The College Board and Educational Testing Service (ETS) are dedicated to the principle of equal opportunity, and their programs, services, and employment policies are guided by that principle.

The College Board is a national nonprofit membership association dedicated to preparing, inspiring, and connecting students to college and opportunity. Founded in 1900, the association is composed of more than 3,900 schools, colleges, universities, and other educational organizations. Each year, the College Board serves over three million students and their parents, 22,000 high schools, and 3,500 colleges, through major programs and services in college admission, guidance, assessment, financial aid, enrollment, and teaching and learning. Among its best-known programs are the SAT®, the PSAT/NMSQT™, the Advanced Placement Program® (AP®), and Pacesetter®. The College Board is committed to the principles of equity and excellence, and that commitment is embodied in all of its programs, services, activities, and concerns.

Copyright © 2001 by College Entrance Examination Board. All rights reserved. College Board, Advanced Placement Program, AP, and the acorn logo are registered trademarks of the College Entrance Examination Board.

1. A *mode* is a value in an array that is larger than both the value immediately before it in the array and the value immediately after it. In other words, a mode occurs at index `k` in the array `A` if `A[k] > A[k - 1]` and `A[k] > A[k + 1]`. The array is *unimodal* if the values increase until they reach a mode, then decrease, so that there is only one mode. For example, the array `A` shown below is unimodal with its mode occurring at index 4. Assume that the mode does not occur at the first or last entry in the array.

Index k	A[k]	
0	3	
1	5	
2	9	
3	10	
4	12	← mode
5	11	
6	9	
7	4	

(a) Write function `IsMode`, as started below. `IsMode` returns `true` if `data[k]` is larger than `data[k - 1]` and larger than `data[k + 1]`; otherwise, it returns `false`. In the example above, the call `IsMode(A, 4)` returns `true` and the call `IsMode(A, 5)` returns `false`.

Complete function `IsMode` below.

```
bool IsMode(const apvector<int> & data, int k)
// precondition:  0 < k < data.length() - 1
{
  return((data[k]>data[k-1])&&(data[k]>data[k+1]));
}
```

A1

A

(b) Write function `ModeIndex`, as started below. `ModeIndex` returns the index of the mode of `data`. You may assume that `data` is unimodal and the mode occurs at an index `k`, where `0 < k < data.length() - 1`. In the example above, the call `ModeIndex(A)` returns `4`.

In writing `ModeIndex`, you may call function `IsMode` specified in part (a). Assume that `IsMode` works as specified, regardless of what you wrote in part (a).

Complete function `ModeIndex` below.

```
int ModeIndex(const apvector<int> & data)
// precondition:  data is unimodal and data.length() ≥ 3
{
   int k;
   for (k=1; k<data.length()-1; k++)
   {
      if (IsMode(data,k))
         return(k);
   }
}
```

Part (c) begins on page 6.

A1

A

Complete function PrintHistogram below.

```
void PrintHistogram(const apvector<int> & data,
                    int longestBar, char barChar)
// precondition:  data is unimodal and data.length() ≥ 3;
//                data[k] ≥ 0 for 0 ≤ k < data.length()
{
   double c; int k, j, numChars;
   c = double(longestBar)/data[ModeIndex(data)];

   for (k=0 ; k< data.length() ; k++)
   {
numChars= data[k]*c;
  for (j=0; j<numChars; j++)
  {
    cout<<barChar;
  }
  cout<<endl;
   }
}
```

A1 B

1. A *mode* is a value in an array that is larger than both the value immediately before it in the array and the value immediately after it. In other words, a mode occurs at index `k` in the array `A` if `A[k] > A[k - 1]` and `A[k] > A[k + 1]`. The array is *unimodal* if the values increase until they reach a mode, then decrease, so that there is only one mode. For example, the array `A` shown below is unimodal with its mode occurring at index `4`. Assume that the mode does not occur at the first or last entry in the array.

Index k	A[k]	
0	3	
1	5	
2	9	
3	10	
4	12	← mode
5	11	
6	9	
7	4	

(a) Write function `IsMode`, as started below. `IsMode` returns `true` if `data[k]` is larger than `data[k - 1]` and larger than `data[k + 1]`; otherwise, it returns `false`. In the example above, the call `IsMode(A, 4)` returns `true` and the call `IsMode(A, 5)` returns `false`.

Complete function `IsMode` below.

```
bool IsMode(const apvector<int> & data, int k)
// precondition:  0 < k < data.length() - 1
```

```
bool IsMode (const apvector <int> & data, int k)
{
    for (k=1 ; k < data.length()-1 ; k++)
        {
            if (data [k] > data [k-1] && data [k] >
                    data [k+1])
                return true;
        }
    return false;
}
```

A1 B

(b) Write function `ModeIndex`, as started below. `ModeIndex` returns the index of the mode of `data`. You may assume that `data` is unimodal and the mode occurs at an index `k`, where `0 < k < data.length() - 1`. In the example above, the call `ModeIndex(A)` returns `4`.

In writing `ModeIndex`, you may call function `IsMode` specified in part (a). Assume that `IsMode` works as specified, regardless of what you wrote in part (a).

Complete function `ModeIndex` below.

```
int ModeIndex(const apvector<int> & data)
// precondition:  data is unimodal and data.length() ≥ 3
```

```
int ModeIndex (const apvector <int> & data)
   {
     int k;
     for (k=1; k< data.length()-1; k++)
        {
          if (IsMode(A,k) == true)
              return k;
        }
   }
```

Part (c) begins on page 6.

A1

B

Complete function PrintHistogram below.

```
void PrintHistogram(const apvector<int> & data,
                    int longestBar, char barChar)
// precondition:  data is unimodal and data.length() ≥ 3;
//                data[k] ≥ 0 for 0 ≤ k < data.length()
```

```
void PrintHistogram (const apvector <int> & data, int longestBar, char barChar)
  {
    int maxBars;
    for (int k=0; k < data.length(); k++)
      {
        maxBars = (longestBar / data [ModeIndex (data)]) * data [k];
        for (int j=0; j <= maxBars; j++)
            cout << barChar;
            endl;
      }
  }
```

A1 C

1. A *mode* is a value in an array that is larger than both the value immediately before it in the array and the value immediately after it. In other words, a mode occurs at index `k` in the array `A` if `A[k] > A[k - 1]` and `A[k] > A[k + 1]`. The array is *unimodal* if the values increase until they reach a mode, then decrease, so that there is only one mode. For example, the array `A` shown below is unimodal with its mode occurring at index `4`. Assume that the mode does not occur at the first or last entry in the array.

Index k	A[k]	
0	3	
1	5	
2	9	
3	10	
4	12	← mode
5	11	
6	9	
7	4	

(a) Write function `IsMode`, as started below. `IsMode` returns `true` if `data[k]` is larger than `data[k - 1]` and larger than `data[k + 1]`; otherwise, it returns `false`. In the example above, the call `IsMode(A, 4)` returns `true` and the call `IsMode(A, 5)` returns `false`.

Complete function `IsMode` below.

```
bool IsMode(const apvector<int> & data, int k)
// precondition:  0 < k < data.length() - 1
{
   for (int i = 0; i < data.length(); i++)
   {
      if (data[k] < data[k-1] && data[k] > data[k+1])
      {  k++; return true;
      }
      return false;
   }
```

A1 C

(b) Write function `ModeIndex`, as started below. `ModeIndex` returns the index of the mode of `data`. You may assume that `data` is unimodal and the mode occurs at an index `k`, where `0 < k < data.length() - 1`. In the example above, the call `ModeIndex(A)` returns 4.

In writing `ModeIndex`, you may call function `IsMode` specified in part (a). Assume that `IsMode` works as specified, regardless of what you wrote in part (a).

Complete function `ModeIndex` below.

```
int ModeIndex(const apvector<int> & data)
// precondition:  data is unimodal and data.length() ≥ 3
  {
      for(int k=0; k< data.length; k++)
      {
          if (IsMode(data, k))
              return k
       }
        else  return -1;
  }
```

Part (c) begins on page 6.

A1

C

Complete function PrintHistogram below.

```
void PrintHistogram(const apvector<int> & data,
                    int longestBar, char barChar)
// precondition:  data is unimodal and data.length() ≥ 3;
//                data[k] ≥ 0 for 0 ≤ k < data.length()
{
    for(int i =0; i < data.length(); i++)
    {  if( IsMode(data, i) )
       {
       longestBar = ModeIndex(data);
       for( int k=0; k <= longestBar ; k++)
        {
           cout << "x";
        }cout<<endl;
       }
       else
       {
         for(int p=0; p <= (20/data[p]); p++
          {
              cout << "x";
          }
          cout <<endl;
       }
    }
}
```