AP Computer Science A
2000 Student Samples

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.

A4/AB1 A

(a) Write member function `GetCoordinates`, as started below. `GetCoordinates` takes a given letter or digit and returns its row and column in the 2-dimensional array. Assume that the parameter `ch` is a capital letter in the range `'A'` through `'Z'` or a digit in the range `'0'` through `'9'`.

The following example shows the point locations of character `ch` in the given matrix.

Encryptor.myMat						ch	Point coordinates	
S	T	U	V	W	X	P	row = 5	col = 3
Y	Z	0	1	2	3	8	row = 2	col = 4
4	5	6	7	8	9	M	row = 5	col = 0
A	B	C	D	E	F			
G	H	I	J	K	L			
M	N	O	P	Q	R			

Complete function `GetCoordinates` below.

```
Point Encryptor::GetCoordinates(char ch) const
// precondition:  'A' ≤ ch ≤ 'Z' or '0' ≤ ch ≤ '9'
// postcondition: returns the row and column number of the
//                location of ch in myMat
{  Point next;
   for(int i=0; i< myMat.NumRows(); i++)
   {
      for(int z=0; z<myMat.NumCols(); z++)
      {
         if(myMat[i][z] == ch)
         {
            next.row = i;
            next.col = z;
         }
      }
   }
   return next;
}
```

Part (b) begins on page 6.

Complete function EncryptTwo below.

```
apstring Encryptor::EncryptTwo(const apstring & pair) const
// precondition:  pair.length() is 2
// postcondition: returns an encoded two-character string
{
    apstring next = " ";
    Point p = GetCoordinates(pair[0]);
    Point p1 = GetCoordinates(pair[1]);
    if ((p.row == p1.row) || (p.col == p1.col))
    {
        next += pair[1];
        next += pair[0];
    }
    else
    {
        next += myMat[p.row][p1.col];
        next += myMat[p1.row][p.col];
    }
    return next;
}
```

Part (c) begins on page 8.

A4/AB1

A

Complete function EncryptWord below.

```
apstring Encryptor::EncryptWord(const apstring & word) const
// precondition:  word contains only capital letters 'A' through 'Z'
//                and digits '0' through '9'.
// postcondition: returns an encrypted version of word, in which every
//                two letters have been examined and encrypted by
//                replacing the original letters with those located
//                in the opposite corners of the rectangle formed by
//                the two letters. If the original word contains an odd
//                number of letters, the last letter is left unchanged.
{
   apstring next = "";
   for(int i=0; i < word.length(); i+=2)
   {
      if(i+1 < word.length())
      {
         next += EncryptTwo(word.substr(i,2));
      }
      else
         next += word[i];
   }
   return next;
}
```

A4/AB1

B

(a) Write member function `GetCoordinates`, as started below. `GetCoordinates` takes a given letter or digit and returns its row and column in the 2-dimensional array. Assume that the parameter `ch` is a capital letter in the range `'A'` through `'Z'` or a digit in the range `'0'` through `'9'`.

The following example shows the point locations of character `ch` in the given matrix.

Encryptor.myMat						ch	Point coordinates	
S	T	U	V	W	X	P	row = 5	col = 3
Y	Z	0	1	2	3	8	row = 2	col = 4
4	5	6	7	8	9	M	row = 5	col = 0
A	B	C	D	E	F			
G	H	I	J	K	L			
M	N	O	P	Q	R			

Complete function `GetCoordinates` below.

```
Point Encryptor::GetCoordinates(char ch) const
// precondition:  'A' ≤ ch ≤ 'Z' or '0' ≤ ch ≤ '9'
// postcondition: returns the row and column number of the
//                location of ch in myMat
    {
for(int r=0;r<6;r++)
   for(int c=0;c<6;c++)
     {
     if(myMat[c][r]==ch)
        {
        Point result(c,r);
        return result;
        }
     }
}
```

Part (b) begins on page 16.

A4/AB1

B

Complete function EncryptTwo below.

```
apstring Encryptor::EncryptTwo(const apstring & pair) const
// precondition:  pair.length() is 2
// postcondition: returns an encoded two-character string
   {
   Point a, b;
   apstring result;
   char  c1, c2;
   c1= pair[0];
   c2=pair[1];
   a= Encrypter. GetCoordinates(c1);
   b= Encrypter. GetCoordinates(c2);
   if((a.row  = b.row )||(a.col= b.col))
       {
       result[0]=pair[1];
       result[1]=pair[0];
       }
    else
       {
       result[0]=myMat[b.col][b.row];
       result[1]=myMat[a.col][a.row];
       }
   return result;
}
```

Part (c) begins on page 18.

A4/AB 1

B

Complete function `EncryptWord` below.

```
apstring Encryptor::EncryptWord(const apstring & word) const
// precondition:  word contains only capital letters 'A' through 'Z'
//                and digits '0' through '9'.
// postcondition: returns an encrypted version of word, in which every
//                two letters have been examined and encrypted by
//                replacing the original letters with those located
//                in the opposite corners of the rectangle formed by
//                the two letters. If the original word contains an odd
//                number of letters, the last letter is left unchanged.
    {
    int len,k=0;
    apstring temp, enctemp, result;
    len = word.length();
    if (len%2!=0)
      len--;
    while (k < len)
       {
       temp = word.substr(k,2);
       enctemp = Encryptor.EncryptTwo(temp);
       result += enctemp;
       k = k+2;
       }
    if(len%2!=0)
      result += word[len];
    return result;
    }
```

A4/AB1

C

(a) Write member function `GetCoordinates`, as started below. `GetCoordinates` takes a given letter or digit and returns its row and column in the 2-dimensional array. Assume that the parameter `ch` is a capital letter in the range `'A'` through `'Z'` or a digit in the range `'0'` through `'9'`.

The following example shows the point locations of character `ch` in the given matrix.

Encryptor.myMat						ch	Point coordinates
S	T	U	V	W	X	P	row = 5 col = 3
Y	Z	0	1	2	3	8	row = 2 col = 4
4	5	6	7	8	9	M	row = 5 col = 0
A	B	C	D	E	F		
G	H	I	J	K	L		
M	N	O	P	Q	R		

Complete function `GetCoordinates` below.

```
Point Encryptor::GetCoordinates(char ch) const
// precondition:  'A' ≤ ch ≤ 'Z' or '0' ≤ ch ≤ '9'
// postcondition: returns the row and column number of the
//                location of ch in myMat
{
  int i, j;
  for (i=0; i < 6; i++)
    for (j=0; j < 6; j++)
      if (myMat[i][j] == ch)
        return (GetCoordinates(ch));
}
```

Part (b) begins on page 16.

A4/AB1

C

Complete function EncryptTwo below.

```
apstring Encryptor::EncryptTwo(const apstring & pair) const
// precondition:  pair.length() is 2
// postcondition: returns an encoded two-character string
{
   int i;
Point pos1 = GetCoordinates(pair[0]);
Point pos2 = GetCoordinates(pair[1]);
  if ((pos1[0] == pos2[0]) || (pos1[1] == pos2[1]))
  {
    pos1 = pos2;
    pos2 = GetCoordinates(pair[0]);
    return (pair[1] + pair[0]);
  }
```

Part (c) begins on page 18.

A4/AB1

C

Complete function `EncryptWord` below.

```
apstring Encryptor::EncryptWord(const apstring & word) const
// precondition:  word contains only capital letters 'A' through 'Z'
//                and digits '0' through '9'.
// postcondition: returns an encrypted version of word, in which every
//                two letters have been examined and encrypted by
//                replacing the original letters with those located
//                in the opposite corners of the rectangle formed by
//                the two letters. If the original word contains an odd
//                number of letters, the last letter is left unchanged.
{
    int i;
    apstring temp, newword = "";

    for (i=0; i < word.length(); i++)
    {
        temp = word[i] + word[i+1];
        ~~newword += temp;~~
        temp = EncryptTwo(temp);
        newword += temp;
    }
    return (newword);
}
```