AP® Computer Science A
2001 Sample Student Responses

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.

(a) Write the `Station` member function `ResetAll`, as started below. `ResetAll` changes the number of gallons sold at each pump to 0.0.

In writing `ResetAll`, you may call any of the public member functions of the `Pump` and `Station` classes. Assume that all these functions work as specified.

Complete function `ResetAll` below.

```
void Station::ResetAll()
// postcondition: For every Pump p in this station
//                   p.GallonsSold() is 0.0
```

```
void Station::ResetAll()
{
    for(int x = 0; x < myPumps.length(); x++)
        myPumps[x].ResetGallonsSold();
}
```

(b) Write the `Station` member function `TotalSales`, as started below. `TotalSales` returns the total cash value of the gallons sold at all pumps. Recall that self-service pumps charge the base price for each gallon of gas, while full-service pumps (pumps 0 and 1) charge $0.25 more per gallon.

In writing `TotalSales`, you may call any of the public member functions of the `Pump` and `Station` classes. Assume that all these functions work as specified.

Complete function `TotalSales` below.

```
double Station::TotalSales() const
// postcondition: returns the total cash value of sales
//                for all pumps
double Station::TotalSales( ) const
{
    double sum = 0.0;
    double x = myBasePrice + 0.25;
    for (int a=0; a<2; a++)
        sum+=(x * myPumps[a].GallonsSold( ));
    for(int k=2;k< myPumps.length( );k++)
        sum+=(myBasePrice* myPumps[k].GallonsSold( ));
    return sum;
}
```

(c) Write the `Station` member function `CloseStation`, as started below. `CloseStation` will write the total amount of money earned from the day's gas sales to the output stream, `logFile`, and then reset the number of gallons sold at each individual pump to 0.0.

In writing `CloseStation`, you may call any of the public member functions of the `Pump` and `Station` classes. Assume that these functions, including `ResetAll` and `TotalSales`, work as specified, regardless of what you wrote in parts (a) and (b).

Complete function `CloseStation` below.

```
void Station::CloseStation(ostream & logFile)
// precondition:  logFile is open and ready for writing
// postcondition: writes the total cash value of
//                all gas sold for the day to logFile;
//                for every Pump p in this station
//                p.GallonsSold() is 0.0
```

```
void Station::CloseStation(ostream & logFile)
{
    logFile << TotalSales( );
    ResetAll( );
}
```

(a) Write the `Station` member function `ResetAll`, as started below. `ResetAll` changes the number of gallons sold at each pump to 0.0.

In writing `ResetAll`, you may call any of the public member functions of the `Pump` and `Station` classes. Assume that all these functions work as specified.

Complete function `ResetAll` below.

```
void Station::ResetAll()
// postcondition: For every Pump p in this station
//                p.GallonsSold() is 0.0
{
  Pump p;
  int k;
  for(k=0; k<myPumps.length; k++)
    p[k].GallonsSold( ) = 0.0;
}
```

(b) Write the `Station` member function `TotalSales`, as started below. `TotalSales` returns the total cash value of the gallons sold at all pumps. Recall that self-service pumps charge the base price for each gallon of gas, while full-service pumps (pumps 0 and 1) charge $0.25 more per gallon.

In writing `TotalSales`, you may call any of the public member functions of the `Pump` and `Station` classes. Assume that all these functions work as specified.

Complete function `TotalSales` below.

```
double Station::TotalSales() const
// postcondition: returns the total cash value of sales
//                for all pumps
{
    double total, fulltot, selftot;
    Pump p;
    int k;
    for(k=0; k<myPumps.length(); k++)
    {
      if((p[k] == p[0])||(p[k] == p[1]))
      {
        myBasePrice += .25;
        fulltot = myBasePrice*(p[k].GallonsSold());
      }
      else
      {
        selftot = myBasePrice*(p[k].GallonsSold());
      }
    }

    total = selftot + fulltot;
    return total;
}
```

(c) Write the `Station` member function `CloseStation`, as started below. `CloseStation` will write the total amount of money earned from the day's gas sales to the output stream, `logFile`, and then reset the number of gallons sold at each individual pump to 0.0.

In writing `CloseStation`, you may call any of the public member functions of the `Pump` and `Station` classes. Assume that these functions, including `ResetAll` and `TotalSales`, work as specified, regardless of what you wrote in parts (a) and (b).

Complete function `CloseStation` below.

```
void Station::CloseStation(ostream & logFile)
// precondition:  logFile is open and ready for writing
// postcondition: writes the total cash value of
//                all gas sold for the day to logFile;
//                for every Pump p in this station
//                p.GallonsSold() is 0.0
{
```

logFile << TotalSales( );

ostream >> logFile ;

ResetAll( );

}

(a) Write the `Station` member function `ResetAll`, as started below. `ResetAll` changes the number of gallons sold at each pump to 0.0.

In writing `ResetAll`, you may call any of the public member functions of the `Pump` and `Station` classes. Assume that all these functions work as specified.

Complete function `ResetAll` below.

```
void Station::ResetAll()
// postcondition: For every Pump p in this station
//                  p.GallonsSold() is 0.0
{
   Pump p;
   int h;
   for(h=0; h<=p.length(); h++)
   {
     p[h].GallonsSold() = 0.0;
   }
}
```

(b) Write the `Station` member function `TotalSales`, as started below. `TotalSales` returns the total cash value of the gallons sold at all pumps. Recall that self-service pumps charge the base price for each gallon of gas, while full-service pumps (pumps 0 and 1) charge $0.25 more per gallon.

In writing `TotalSales`, you may call any of the public member functions of the `Pump` and `Station` classes. Assume that all these functions work as specified.

Complete function `TotalSales` below.

```
double Station::TotalSales() const
// postcondition: returns the total cash value of sales
//                for all pumps
{
  Pump P;
  if(p[0] || p[1])
   {
     return (myBasePrice + .25);
   }
  else
  {
    return myBasePrice;
  }
}
```

(c) Write the `Station` member function `CloseStation`, as started below. `CloseStation` will write the total amount of money earned from the day's gas sales to the output stream, `logFile`, and then reset the number of gallons sold at each individual pump to 0.0.

In writing `CloseStation`, you may call any of the public member functions of the `Pump` and `Station` classes. Assume that these functions, including `ResetAll` and `TotalSales`, work as specified, regardless of what you wrote in parts (a) and (b).

Complete function `CloseStation` below.

```
void Station::CloseStation(ostream & logFile)
// precondition:  logFile is open and ready for writing
// postcondition: writes the total cash value of
//                all gas sold for the day to logFile;
//                for every Pump p in this station
//                p.GallonsSold() is 0.0
{
  Pump p;
  ifstream in;
  while(ResetAll())
  {
    in >> TotalSales();
  }
}
```