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.

A2 A

2. This question involves reasoning about the code from the Large Integer Case Study. A copy of the code is provided as part of this exam.

   (a) Write the new `BigInt` public member function `IsOdd`, as started below. `IsOdd` should return `true` if the `BigInt` is odd; otherwise, it should return `false`.

   You may NOT assume that the `%` or `%=` operators have been defined for the `BigInt` class.

   Complete function `IsOdd` below.

```
bool BigInt::IsOdd() const
// postcondition: returns true if this BigInt is odd;
//                otherwise, returns false
{
  return (GetDigit(0) % 2) == 1;
} // the ==1 isn't really necessary, but it's clearer.
```

A2 A

(b) Write the free function `Power`, as started below. `Power` returns the value of `base` to the `exp` power, that is $base^{exp}$, where `exp` ≥ 0. For example, the call `Power(3, 5)` returns 243, which is 3^5.

You must use the following algorithm.

> Initialize a variable, `product`, to be 1.
> While `exp` is not zero do the following:
> if `exp` is odd, `product` is set to `product` times the `base`
> square the `base`
> divide `exp` by two
> When done, `product` contains the result.

Assume that a new member function, `DivBy2`, has been defined for the `BigInt` class, as specified below. `DivBy2` divides this `BigInt` by 2 (using integer division). (You do not need to write the body of `DivBy2`.)

```
void BigInt::DivBy2();  // this BigInt is divided by 2
```

In writing `Power`, you may use the `BigInt` public member function `DivBy2` specified above and you may use the `BigInt` public member function `IsOdd` specified in part (a). Assume that `IsOdd` works as specified, regardless of what you wrote in part (a).

Complete function `Power` below.

```
BigInt Power(const BigInt & base, const BigInt & exp)
// precondition:  base > 0 and exp ≥ 0
// postcondition: returns the value of base to the exp
{
  BigInt product(1);
  BigInt tbase(base);
  BigInt texp(exp);
  while(texp != 0)
  {
    if (texp.IsOdd())
    { product *= tbase;
    }
   tbase *= tbase;
   texp.DivBy2();
  }
 return product;
}
```

A2_

B

2. This question involves reasoning about the code from the Large Integer Case Study. A copy of the code is provided as part of this exam.

   (a) Write the new `BigInt` public member function `IsOdd`, as started below. `IsOdd` should return `true` if the `BigInt` is odd; otherwise, it should return `false`.

   You may NOT assume that the `%` or `%=` operators have been defined for the `BigInt` class.

   Complete function `IsOdd` below.

```
bool BigInt::IsOdd() const
// postcondition: returns true if this BigInt is odd;
//                otherwise, returns false
{
  int n;
  n= GetDigit(0);
  if (n==1 || n== 3 || n== 5 || n== 7|| n==9)
      return true;
  else
      return false;
}
```

A2 B

(b) Write the free function `Power`, as started below. `Power` returns the value of `base` to the `exp` power, that is $base^{exp}$, where `exp` ≥ 0. For example, the call `Power(3, 5)` returns 243, which is 3^5.

You must use the following algorithm.

> Initialize a variable, `product`, to be 1.
> While `exp` is not zero do the following:
> - if `exp` is odd, `product` is set to `product` times the `base`
> - square the `base`
> - divide `exp` by two
>
> When done, `product` contains the result.

Assume that a new member function, `DivBy2`, has been defined for the `BigInt` class, as specified below. `DivBy2` divides this `BigInt` by 2 (using integer division). (You do not need to write the body of `DivBy2`.)

```
void BigInt::DivBy2();  // this BigInt is divided by 2
```

In writing `Power`, you may use the `BigInt` public member function `DivBy2` specified above and you may use the `BigInt` public member function `IsOdd` specified in part (a). Assume that `IsOdd` works as specified, regardless of what you wrote in part (a).

Complete function `Power` below.

```
BigInt Power(const BigInt & base, const BigInt & exp)
// precondition:  base > 0 and exp ≥ 0
// postcondition: returns the value of base to the exp
{
int product=1, k;
while (exp != 0)
{
    if (exp.IsOdd())
    {
      product = product * base;
      base *= base;
      exp.DivBy2;
    }
    else                          // if the exp is even
    {
      for (k=0; k<exp; k++)
          base *= base;
      product = base;
      exp=0;
    }
}
return product;
}
```

A2 C

2. This question involves reasoning about the code from the Large Integer Case Study. A copy of the code is provided as part of this exam.

(a) Write the new `BigInt` public member function `IsOdd`, as started below. `IsOdd` should return `true` if the `BigInt` is odd; otherwise, it should return `false`.

You may NOT assume that the `%` or `%=` operators have been defined for the `BigInt` class.

Complete function `IsOdd` below.

```
bool BigInt::IsOdd() const
// postcondition: returns true if this BigInt is odd;
//                otherwise, returns false
{
  if(BigInt/2 == 0)
   return false;
  else
   return true;
}
```

A2 C

(b) Write the free function `Power`, as started below. `Power` returns the value of `base` to the `exp` power, that is $base^{exp}$, where `exp` $\geq$ `0`. For example, the call `Power(3, 5)` returns `243`, which is 3^5.

You must use the following algorithm.

> Initialize a variable, `product`, to be `1`.
> While `exp` is not zero do the following:
> if `exp` is odd, `product` is set to `product` times the `base`
> square the `base`
> divide `exp` by two
> When done, `product` contains the result.

Assume that a new member function, `DivBy2`, has been defined for the `BigInt` class, as specified below. `DivBy2` divides this `BigInt` by `2` (using integer division). (You do not need to write the body of `DivBy2`.)

```
void BigInt::DivBy2();  // this BigInt is divided by 2
```

In writing `Power`, you may use the `BigInt` public member function `DivBy2` specified above and you may use the `BigInt` public member function `IsOdd` specified in part (a). Assume that `IsOdd` works as specified, regardless of what you wrote in part (a).

Complete function `Power` below.

```
BigInt Power(const BigInt & base, const BigInt & exp)
// precondition:  base > 0 and exp ≥ 0
// postcondition: returns the value of base to the exp
```

```
{
 int product=1;
 while(exp!=0)
  {if(IsOdd=="true")
   {product*=base;
    base*=base;
    exp/=2;
   }
  }
  return product;
}
```

GO ON TO THE NEXT PAGE.