AP® Computer Science AB
2002 Sample Student Responses

The materials included in these files are intended for use by AP teachers for course and exam preparation in the classroom; 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.

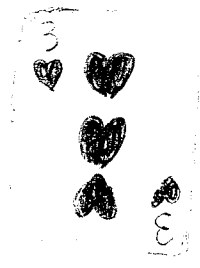
The College Board is a national nonprofit membership association dedicated to preparing, inspiring, and connecting students to college and opportunity. Founded in 1900, the association is composed of more than 4,200 schools, colleges, universities, and other educational organizations. Each year, the 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®, and the Advanced Placement Program® (AP®). 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.

(a) You will write free function `AppendQueue`, which is described as follows. `AppendQueue` should remove all the cards from the parameter `source` and add them to the parameter `destination` in the same order.

Complete function `AppendQueue` below.

```
void AppendQueue(apqueue<Card> & destination,
                 apqueue<Card> & source)
// postcondition: all cards have been removed from source
//                and added to destination in the same order
{
   Card temp;

   while (!source.isEmpty())
   {
      source.dequeue(temp);
      destination.enqueue(temp);
   }
}
```



Complete function `OneRound` below.

```
void OneRound(apqueue<Card> & pile1,
              apqueue<Card> & pile2)
// precondition:  pile1.length() > 0;  pile2.length() > 0
// postcondition: pile1 and pile2 have been updated according to the
//                rules of the game
{
   Card card1, card2;
   apqueue<Card> discard;
   do
   {
      pile1.dequeue(card1);
      pile2.dequeue(card2);

      discard.enqueue(card1);
      discard.enqueue(card2);
   } while (!pile1.isEmpty() && !pile2.isEmpty() && card1 == card2);
   if (card1.Value > card2.Value)
      AppendQueue(pile1, discard);
   else if (card2.Value > card1.Value)
      AppendQueue(pile2, discard);
   else if (pile1.isEmpty() && !pile2.isEmpty())
      AppendQueue(pile2, discard);
   else if (pile2.isEmpty() && !pile1.isEmpty())
      AppendQueue(pile1, discard);
}
```

GO ON TO THE NEXT PAGE.

AB 2 B

(a) You will write free function `AppendQueue`, which is described as follows. `AppendQueue` should remove all the cards from the parameter `source` and add them to the parameter `destination` in the same order.

Complete function `AppendQueue` below.

```
void AppendQueue(apqueue<Card> & destination,
                 apqueue<Card> & source)
// postcondition: all cards have been removed from source
//                and added to destination in the same order
{
    Card moveCard;

    while (!source.isEmpty()) {
        source.dequeue(moveCard);
        destination.enqueue(moveCard);
    }

}
```

GO ON TO THE NEXT PAGE.

Complete function OneRound below.

```
void OneRound(apqueue<Card> & pile1,
              apqueue<Card> & pile2)
// precondition:  pile1.length() > 0;  pile2.length() > 0
// postcondition: pile1 and pile2 have been updated according to the
//                rules of the game
{
    int one, two;
    Card first, second;
    apqueue<Card>  discard;

    if (pile1.isEmpty() && pile2.isEmpty())
        return;
    while (!pile1.isEmpty() || !pile2.isEmpty()) {  // some card in either
       if (pile1.isEmpty()){
           AppendQueue(pile2, discard);
           return;
       }
       if (pile2.isEmpty()){
          AppendQueue(pile1, discard);
          return;
       }
       pile1.dequeue(first);
       pile2.dequeue(second);

       if (first.value() == second.value()){
           discard.enqueue(first);
           discard.enqueue(second);
       }
       else if (first.value() > second.value()){
           AppendQueue(pile1, discard);
           pile1.enqueue(first);
           pile1.enqueue(second);
           return;
       }
       else {
          AppendQueue(pile2, discard);
          pile2.enqueue(first);
          pile2.enqueue(second);
          return;
       }
    } // end while
    return; // at this pt, both queues are empty
} // end fxn
```

GO ON TO THE NEXT PAGE.

AB 2 C

(a) You will write free function `AppendQueue`, which is described as follows. `AppendQueue` should remove all the cards from the parameter `source` and add them to the parameter `destination` in the same order.

Complete function `AppendQueue` below.

```
void AppendQueue(apqueue<Card> & destination,
                 apqueue<Card> & source)
// postcondition: all cards have been removed from source
//                and added to destination in the same order
{  Card temp;

   While(!(source.IsEmpty()))
    {
      source.dequeue(temp);
      destination.enqueue(temp);

    }

}
```

GO ON TO THE NEXT PAGE.

Complete function `OneRound` below.

```
void OneRound(apqueue<Card> & pile1,
              apqueue<Card> & pile2)
// precondition:  pile1.length() > 0;  pile2.length() > 0
// postcondition: pile1 and pile2 have been updated according to the
//                rules of the game

{Card temp1, temp2, temp3;
 apqueue<Card> discard;

 while ((pile1.length() != 0) || (pile2.length() != 0))
 {

     pile1.dequeue(temp1);
     pile2.dequeue(temp2);
     if (card.temp1.value == card.temp2.value)
      {
         discard.enqueue(temp1);
         discard.enqueue(temp2);
      }
     else if (card.temp1.value > card.temp2.value)
      {
        pile1.enqueue(temp1);
        pile1.enqueue(temp2);
      }
     else
      {
        pile2.enqueue(temp1);
        pile2.enqueue(temp2);
      }
 }
```

AppendQueue(pile1, discard);

AppendQueue(pile2, discard);

GO ON TO THE NEXT PAGE.