

AP® Computer Science A
2007 Free-Response Questions

The College Board: Connecting Students to College Success

The College Board is a not-for-profit membership association whose mission is to connect students to college success and opportunity. Founded in 1900, the association is composed of more than 5,000 schools, colleges, universities, and other educational organizations. Each year, the College Board serves seven million students and their parents, 23,000 high schools, and 3,500 colleges through major programs and services in college admissions, 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 excellence and equity, and that commitment is embodied in all of its programs, services, activities, and concerns.

© 2007 The College Board. All rights reserved. College Board, Advanced Placement Program, AP, AP Central, SAT, and the acorn logo are registered trademarks of the College Board. PSAT/NMSQT is a registered trademark of the College Board and National Merit Scholarship Corporation.
Permission to use copyrighted College Board materials may be requested online at:
www.collegeboard.com/inquiry/cbpermit.html.

Visit the College Board on the Web: www.collegeboard.com.
AP Central is the official online home for the AP Program: apcentral.collegeboard.com.

COMPUTER SCIENCE A
SECTION II

Time—1 hour and 45 minutes
Number of questions—4
Percent of total grade—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:
- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

1. A positive integer is called a "self-divisor" if every decimal digit of the number is a divisor of the number, that is, the number is evenly divisible by each and every one of its digits. For example, the number 128 is a self-divisor because it is evenly divisible by 1, 2, and 8. However, 26 is not a self-divisor because it is not evenly divisible by the digit 6. Note that 0 is not considered to be a divisor of any number, so any number containing a 0 digit is NOT a self-divisor. There are infinitely many self-divisors.

```
public class SelfDivisor
{
  /** @param number the number to be tested
   *          Precondition: number > 0
   *  @return true if every decimal digit of number is a divisor of number;
   *          false otherwise
   */
  public static boolean isSelfDivisor(int number)
  {  /* to be implemented in part (a) */  }


  /** @param start starting point for values to be checked
   *          Precondition: start > 0
   *  @param num the size of the array to be returned
   *          Precondition: num > 0
   *  @return an array containing the first num integers ≥ start that are self-divisors
   */
  public static int[] firstNumSelfDivisors(int start, int num)
  {  /* to be implemented in part (b) */  }


  // There may be fields, constructors, and methods that are not shown.
}
```

© 2007 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

2. This question involves reasoning about the code from the Marine Biology Simulation case study. A copy of the code is provided as part of this exam.

   A `PounceFish` is a type of fish that looks for prey and then "pounces" on it. A `PounceFish` can see only a limited distance in its forward direction. If the `PounceFish` sees another fish, it rushes forward and eats the nearest one that it sees, ending up in the location where its prey was originally located. If the `PounceFish` does not see another fish, it acts as a `Fish`.

   The `PounceFish` class is shown below.

```
public class PounceFish extends Fish
{
  private int range;   // the distance that a PounceFish can see; range > 0


  /** Looks ahead range locations in current direction
   *  @return the nearest fish in that direction within range (if any);
   *          null if no such fish is found
   */
  private Fish findFish()
  {  /* to be implemented in part (a) */  }


  /** Acts for one step in the simulation
   */
  public void act()
  {  /* to be implemented in part (b) */  }


  // There may be fields, constructors, and methods that are not shown.
}
```

© 2007 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

The following diagrams show an example environment containing a `PounceFish` (represented by `P`) and other fish (represented by `F1`, `F2`, etc.). The direction of the `PounceFish` is indicated by the character `">"` showing that, in this example, the `PounceFish` is facing east. If the `PounceFish` can see 2 or more locations ahead in its forward direction, it will see fish `F3` as shown in the first diagram and will move to that location to eat it, causing `F3` to die as shown in the second diagram.

Environment before the `PounceFish` acts

North

		0	1	2	3	4	5	
	0			F1				
West	1	F2	P>		F3	F4		**East**
	2		F5					
	3							

South

Environment after the `PounceFish` acts

North

		0	1	2	3	4	5	
	0			F1				
West	1	F2			P>	F4		**East**
	2		F5					
	3							

South

If the `PounceFish` in the first diagram above could see only 1 location ahead, it would not see any prey and therefore would act as an ordinary fish.

© 2007 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

(a) Write the `PounceFish` method `findFish`. If any fish are located within `range` locations in the direction that the `PounceFish` is currently facing, the method returns the nearest of these. Otherwise, the method returns `null`.

Complete method `findFish` below.

```
/** Looks ahead range locations in current direction
 *  @return the nearest fish in that direction within range (if any);
 *          null if no such fish is found
 */
private Fish findFish()
```

(b) Override the `act` method for the `PounceFish` class. A `PounceFish` attempts to find a fish that it can eat. If it finds such a fish, the `PounceFish` eats it (causing it to die) and moves to its location. If the `PounceFish` does not find a fish that it can eat, it acts as an ordinary fish.

In writing `act`, assume that `findFish` works as specified, regardless of what you wrote in part (a).

Complete method `act` below.

```
/** Acts for one step in the simulation
 */
public void act()
{
  if ( ! isInEnv() )
    return;

  // Write your code below.
```

© 2007 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).