

## Current and Recent Members of the AP Computer Science Development Committee

**Donald Allen\***

Troy High School  
California

**Scot Drysdale,\*** Chair

Dartmouth College  
New Hampshire

**Reginald A. Hahne\***

Atholton High School  
Maryland

**Cay S. Horstmann\***

San Jose State University  
California

**Judith S. Hromcik**

Arlington High School  
Texas

**Richard G. Kick**

Hindsdale Central High School  
Illinois

**Joseph W. Kmoch**

Washington High School  
Wisconsin

**Andrea W. Lawrence**

Spelman College  
Georgia

**Christopher H. Nevison,** former Chief

Reader  
Colgate University  
New York

**Ann Shen\***

Bishop Strachan School  
Canada

**David Reed,\*** Chief Reader

Creighton University  
Nebraska

**Mark A. Weiss,** former Chair

Florida International University  
Florida

**Laurie A. White\***

Mercer University  
Georgia

**Julie D. Zelenski**

Stanford University  
California

\* Current committee member

# Setting a Policy for AP<sup>®</sup> Computer Science

The purpose of this guide is to provide college faculty and administrators with research data, participation and performance data of AP<sup>®</sup> Computer Science students, curricular content, and sample exam questions to facilitate the establishment of appropriate credit and placement policies for AP Computer Science.

The Advanced Placement Program<sup>®</sup> (AP) provides motivated students with the opportunity to take college-level courses while still in high school. Students demonstrate their mastery of the curriculum by taking AP Exams—35 exams, including two in Computer Science, are available in 20 subject areas. In 2005, more than 1.2 million students took AP Exams worldwide. Of the 2.1 million AP Exams taken in 2005, more than 19,000 were in Computer Science (14,000 in Computer Science A, and 5,000 in Computer Science AB). More than 3,000 colleges and universities, including many international institutions, accept qualifying AP Exam scores for credit, placement, or both.

Throughout its 50-year history, the AP Program has maintained high standards of rigor in its courses and exams. Since its inception, AP has been a respected force in American education due to the critical involvement of college and university faculty members.

## Computer Science Faculty Involvement in AP

College and university faculty members play a vital role in every stage of development and scoring of an AP course and exam, helping to ensure their high quality. Each AP discipline has its own Development Committee—composed of college and university professors and experienced AP teachers—that is responsible for creating the course guidelines and exam questions. College and university faculty members also serve as the Chief Readers, responsible for establishing the exam-scoring guidelines and overseeing the annual AP Reading of the free-response section for their academic discipline.

“Before I was invited to help write the AP Computer Science Exam I was somewhat prejudiced against it. I didn’t think that a high school Introduction to Computer Science course could compare to what we did at Dartmouth. When I looked at the exam I was surprised by the level of understanding required to answer many of the questions. It was clear that a student who did well on this exam would learn little new in our introductory course and would be much better off taking a more advanced CS course. Furthermore, such a student would have an unfair advantage over the novices for whom the introductory course is designed, and might scare off some of these novices. I suggest that those of you who might share my original opinion look at a released exam.”

—Scot Drysdale, AP Computer Science Development Committee Chair  
Dartmouth College

The College Board publication *AP and Higher Education* discusses the following topics at greater length: how to set an AP policy, AP research studies, the development of AP courses and exams, and the AP Exam scoring. For more information or to request a copy of this publication, please go to [apcentral.collegeboard.com/highered](http://apcentral.collegeboard.com/highered).

## How to Set an AP Policy

The College Board encourages higher education institutions to base their AP policy decisions on data and research, and recognizes that different institutions and departments will set different policies, based upon factors unique to their institution, student body, and academic discipline. The best way for colleges and universities to determine their AP credit and placement policies is to conduct their own research on the performance of AP and non-AP students at their own institution and in their own department.

### Research on AP Computer Science Student Performance

Research studies show that students who do well on an AP Exam are academically prepared to place out of a corresponding college course and move on to the next higher-level course in the discipline. See Tables 1 and 2 for data from a research study comparing AP and non-AP student performance in second-level college computer science courses.

Table 1: Student Performance in Second-Level College Computer Science Courses  
AP Computer Science A Students Versus Non-AP Students

	AP EXAM GRADE	GPA	PERCENT OF STUDENTS SCORING AN A OR B
Students Who Place Out of Intro. Course	AP 5	3.42	87
	AP 4	3.11	83
	AP 3	2.86	64
Students Who Complete Intro. Course	Non-AP	2.65	58

Table 2: Student Performance in Second-Level College Computer Science Courses  
AP Computer Science AB Students Versus Non-AP Students

	AP EXAM GRADE	GPA	PERCENT OF STUDENTS SCORING AN A OR B
Students Who Place Out of Intro. Course	AP 5	3.35	84
	AP 4	3.45	92
	AP 3	3.16	84
	AP 2	2.77	57
Students Who Complete Intro. Course	Non-AP	2.62	57

Taking the AP course and exam stimulates further interest in the subject area and encourages deeper disciplinary knowledge.

Research studies show that students who take the AP Computer Science Exams are significantly more likely to take further course work in computer science than students who do not take the AP Exam. Higher scores on the AP Exam make this trend even more pronounced, with a greater likelihood of majoring or minoring in the discipline. See Tables 3 and 4 for data from this research study.

Table 3: Additional College Computer Science Course Work  
AP Computer Science A Students Versus Non-AP Students

	AP EXAM GRADE	PERCENT TAKING ADDITIONAL COMPUTER SCIENCE COURSES	AVERAGE NUMBER OF COLLEGE COMPUTER SCIENCE COURSES TAKEN
AP Computer Science A Students	AP 5	66	3.5
	AP 4	63	2.6
	AP 3	62	2.2
	AP 2	56	1.7
	AP 1	48	1.4
Non-AP Students	Non-AP	32	0.6

Table 4: Additional College Computer Science Course Work  
AP Computer Science AB Students Versus Non-AP Students

	AP EXAM GRADE	PERCENT TAKING ADDITIONAL COMPUTER SCIENCE COURSES	AVERAGE NUMBER OF COLLEGE COMPUTER SCIENCE COURSES TAKEN
AP Computer Science AB Students	AP 5	66	3.7
	AP 4	61	2.9
	AP 3	68	2.5
	AP 2	58	2.3
	AP 1	60	1.7
Non-AP Students	Non-AP	32	0.6

PDF copies of these and other research studies can be found at [apcentral.collegeboard.com/colleges/research](http://apcentral.collegeboard.com/colleges/research).

In addition to research studies on AP student performance, the College Board conducts college comparability studies to measure the degree to which the AP courses and exams are equivalent in content and difficulty to corresponding college courses. The AP Exam scoring rubric is established so that the lowest composite score that earns an AP grade of 5 is equivalent to the average score earned by college students who received grades of A in a comparable course. The lowest score that earns an AP grade of 4 is equivalent to the average B, and the lowest score that earns an AP grade of 3 is equivalent to the average C.

The research that the College Board conducts is intended to help institutions and academic departments as they establish appropriate AP policies. AP Central® ([apcentral.collegeboard.com](http://apcentral.collegeboard.com)), the College Board's online home for AP professionals, contains other resources that may assist in this process, including the Course Description, released exam questions, and sample student responses at different levels of ability.

For more information go to:  
[apcentral.collegeboard.com/compscia/exam](http://apcentral.collegeboard.com/compscia/exam)  
[apcentral.collegeboard.com/compsciab/exam](http://apcentral.collegeboard.com/compsciab/exam)

# AP Computer Science Students, Courses, and Exams

## Participation and Performance Data for AP Computer Science Students in 2005

Total Number of Schools Offering AP Computer Science A: 2,138  
 Total Number of Schools Offering AP Computer Science AB: 1,137

Table 5: AP Computer Science Exam Score Distribution, 2005

EXAM GRADE	NUMBER OF EXAMINEES	% AT
Score of 5	4,086	21.5%
Score of 4	4,230	22.2%
Score of 3	3,020	15.9%
Score of 2	1,913	10.1%
Score of 1	5,772	30.3%
	19,021	100.0%

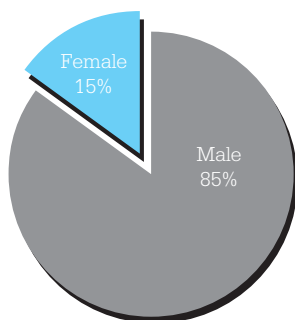
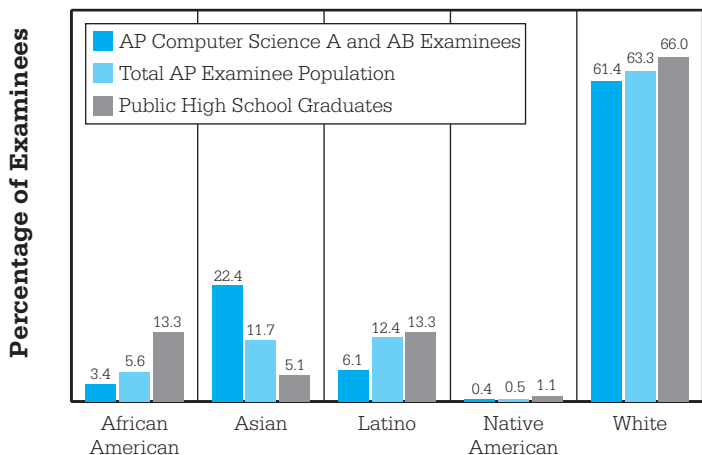


Figure 1: AP Computer Science A and AB by Gender, 2005

Figure 2: AP Computer Science A and AB Examinees by Race and Ethnicity, 2005



## The AP Computer Science Courses

The two AP Computer Science courses are each designed to be the equivalent of a first-semester college-level course in Computer Science. The content of AP Computer Science A is a subset of the content of Computer Science AB; thus AP Computer Science A may be more appropriately considered as equivalent to the introductory college course for noncomputer science majors and AP Computer Science AB as the equivalent intro college course for majors. Both courses emphasize object-oriented programming methodology with a concentration on problem solving and algorithm development and including the study of data structures, design, and abstraction. AP Computer Science AB includes a more formal and in-depth study of algorithms, data structures, design, and abstraction. Both courses use the Java programming language.

The Development Committee creates the guidelines for the AP Computer Science courses and designs the AP Exams. Periodically the Development Committee conducts curriculum surveys, sent to professors who teach the comparable college-level course, that help ensure that the AP Computer Science courses remain current with concepts and themes as taught in college and university classrooms. The Development Committee has created a topic outline that covers the main subject areas that should be taught for each course.

Beginning in fall 2006, AP Computer Science teachers and principals of schools where AP Computer Science is taught must certify that their 2007-08 courses follow all the requirements stipulated by the Development Committee, including using a college-level textbook and providing the computer equipment to complete the course requirements, in order to ensure that the AP course reflects college-level standards. By completing this AP Course Audit, high schools will receive individual licenses to label their computer science courses “AP.” In fall 2007, colleges and universities will receive a list of all high schools authorized to use the “AP” designation for their computer science courses.

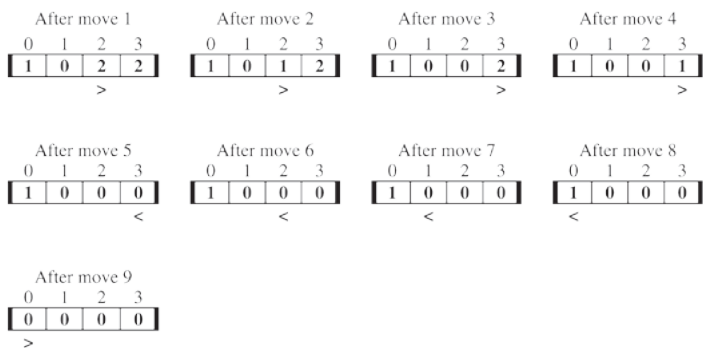
## AP Credit Policy Info on the Web

Information about AP credit and placement policies at more than 1,000 colleges and universities is available on the College Board’s Web site at [www.collegeboard.com/ap/creditpolicy](http://www.collegeboard.com/ap/creditpolicy).

COMPUTER SCIENCE A AND AB	COMPUTER SCIENCE AB ONLY
<b>I. Object-Oriented Program Design</b>	
A. Program design	
1. Read and understand a problem description, purpose, and goals	1. Specify the purpose and goals for a problem
2. Apply data abstraction and encapsulation	
3. Read and understand class specifications and relationships among the classes ("is-a," "has-a" relationships)	3. Decompose a problem into classes; define relationships and responsibilities of those classes
4. Understand and implement a given class hierarchy	
5. Identify reusable components from existing code using classes and class libraries	
B. Class design	
1. Design and implement a class	1. Design and implement a set of interacting classes
2. Design an interface	
3. Choose appropriate data representation and algorithms	3. Choose appropriate advanced data structures and algorithms
4. Apply functional decomposition	
5. Extend a given class using inheritance	
<b>II. Program Implementation</b>	
A. Implementation techniques	
1. Methodology	
a. Object-oriented development	
b. Top-down development	
c. Encapsulation and information hiding	
d. Procedural abstraction	
B. Programming constructs	
1. Primitive types versus objects	
2. Declaration	
a. Constant declarations	
b. Variable declarations	
c. Class declarations	
d. Interface declarations	
e. Method declarations	
f. Parameter declarations	
3. Console output (System.out.print/println)	
4. Control	
a. Methods	
b. Sequential	
c. Conditional	
d. Iteration	
e. Recursion	
C. Java library classes (included in the A-level AP Java Subset)	C. Java library classes (included in the AB-level AP Java Subset)
<b>III. Program Analysis</b>	
A. Testing	
1. Test classes and libraries in isolation	
2. Identify boundary cases and generate appropriate test data	
3. Perform integration testing	
B. Debugging	
1. Categorize errors: compile time, run time, logic	
2. Identify and correct errors	
3. Techniques: use a debugger, add extra output statements, hand-trace code	
C. Understand and modify existing code	
D. Extend existing code using inheritance	
E. Understand error handling	
1. Understand run-time exceptions	
2. Throw run-time exceptions	
F. Reason about programs	
1. Pre- and postconditions	
2. Assertions	

COMPUTER SCIENCE A AND AB	COMPUTER SCIENCE AB ONLY
	3. Invariants
G. Analysis of algorithms	
1. Informal comparisons of running times	
2. Exact calculation of statement execution counts	
	3. Big-Oh notation
	4. Worst-case and average-case time and space analysis
H. Numerical representations and limits	
1. Representations of numbers in different bases	
2. Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error)	
<b>IV. Standard Data Structures</b>	
A. Simple data types (int, boolean, double)	
B. Classes	
C. One-dimensional arrays	
	D. Two-dimensional arrays
	E. Linked lists (singly, doubly, circular)
	F. Stacks
	G. Queues
	H. Trees
	I. Heaps
	J. Priority queues
	K. Sets
	L. Maps
<b>V. Standard Algorithms</b>	
A. Operations on A-level data structures previously listed	A. Operations on AB-level data structures previously listed
1. Traversals	1. Traversals
2. Insertions	2. Insertions
3. Deletions	3. Deletions
	4. Iterators
B. Searching	
1. Sequential	
2. Binary	
	3. Hashing
C. Sorting	
1. Selection	
2. Insertion	
3. Mergesort	
	4. Quicksort
	5. Heapsort
<b>VI. Computing in Context</b>	
A. Major hardware components	
1. Primary and secondary memory	
2. Processors	
3. Peripherals	
B. System software	
1. Language translators/compilers	
2. Virtual machines	
3. Operating systems	
C. Types of systems	
1. Single-user systems	
2. Networks	
D. Responsible use of computer systems	
1. System reliability	
2. Privacy	
3. Legal issues and intellectual property	
4. Social and ethical ramifications of computer use	





After nine moves, the robot stops because the hall is clear.

The PR2004 is modeled by the class `Robot` as shown in the following declaration.

```
public class Robot
{
private int[] hall;
private int pos; // current position(tile number) of Robot
private boolean facingRight; // true means this Robot is facing right

// constructor not shown

// postcondition: returns true if this Robot has a wall immediately in
//                 front of it, so that it cannot move forward;
//                 otherwise, returns false
private boolean forwardMoveBlocked()
{ /* to be implemented in part (a) */ }

// postcondition: one move has been made according to the
//                 specifications above and the state of this
//                 Robot has been updated
private void move()
{ /* to be implemented in part (b) */ }

// postcondition: no more items remain in the hallway;
//                 returns the number of moves made
public int clearHall()
{ /* to be implemented in part (c) */ }

// postcondition: returns true if the hallway contains no items;
//                 otherwise, returns false
private boolean hallIsClear()
{ /* implementation not shown */ }
}
```

In the `Robot` class, the number of items on each tile in the hall is stored in the corresponding entry in the array `hall`. The current position is stored in the instance variable `pos`. The boolean instance variable `facingRight` is `true` if the Robot is facing to the right and is `false` otherwise.

- (a) Write the `Robot` method `forwardMoveBlocked`. Method `forwardMoveBlocked` returns `true` if the robot has a wall immediately in front of it, so that it cannot move forward. Otherwise, `forwardMoveBlocked` returns `false`.

Complete method `forwardMoveBlocked` below.

```
// postcondition: returns true if this Robot has a wall immediately in
//                 front of it, so that it cannot move forward;
//                 otherwise, returns false
private boolean forwardMoveBlocked()
```

- (b) Write the `Robot` method `move`. Method `move` has the robot carry out one move as specified at the beginning of the question. The specification for a move is repeated here for your convenience.

1. If there are any items on the current tile, then one item is removed.
2. If there are more items on the current tile, then the robot remains on the current tile facing the same direction.
3. If there are no more items on the current tile
  - a) if the robot can move forward, it advances to the next tile in the direction that it is facing;
  - b) otherwise, if the robot cannot move forward, it reverses direction and does not change position.

In writing `move`, you may use any of the other methods in the `Robot` class. Assume these methods work as specified, regardless of what you wrote in part (a). Solutions that reimplement the functionality provided by these methods, rather than invoking these methods, will not receive full credit.

Complete method `move` below.

```
// postcondition: one move has been made according to the
//                 specifications above and the state of this
//                 Robot has been updated
private void move()
```

- (c) Write the `Robot` method `clearHall`. Method `clearHall` clears the hallway, repeatedly having this robot make a move until the hallway has no items, and returns the number of moves made.

In the example at the beginning of this problem, `clearHall` would take the robot through the moves shown and return 9, leaving the robot in the state shown in the final diagram.

In writing `clearHall`, you may use any of the other methods in the `Robot` class. Assume these methods work as specified, regardless of what you wrote in parts (a) and (b). Solutions that reimplement the functionality provided by these methods, rather than invoking these methods, will not receive full credit.

Complete method `clearHall` below.

```
// postcondition: no more items remain in the hallway;
//                 returns the number of moves made
public int clearHall()
```

### Question 3 (Computer Science AB)

Consider designing and implementing a set of classes and interfaces to represent books, library items, and library books.

- (a) A library item contains two pieces of information: a unique ID (a `String`) and a holder (also a `String`). If the library item has been checked out, the holder is the name of the person who has checked out this item, otherwise the holder is `null`. The ID of a library item cannot change after it is created, but the holder can change. Write the `LibraryItem` interface that abstracts this functionality.
- (b) A book consists of an author and a title, neither of which can change once the book is created. The `Book` class is represented as follows.

```
public class Book
{
    private String theAuthor;
    private String theTitle;

    public Book(String author, String title)
    { theAuthor = author; theTitle = title; }

    public String getAuthor()
    { return theAuthor; }

    public String getTitle()
    { return theTitle; }
}
```

A library book is a book that is also a library item. Write the complete class `LibraryBook`, implementing the required methods.

- (c) A library is a collection of library items and supports the following operations.
1. Add a library item to the library.
  2. Check out a library item by specifying its ID and new holder.
  3. Determine the current holder of a library item, given its ID.

Consider the following incomplete class declaration.

```
public class Library
{
    private SomeClass items;

    public Library()
    { /* implementation not shown */ }

    public void add(LibraryItem theItem)
    { /* implementation not shown */ }

    public void checkout(String id, String holder)
    { /* implementation not shown */ }

    public String getHolder(String id)
    { /* implementation not shown */ }
}
```

Choose a data representation for `SomeClass` from the `java.util` class library that allows the operations `add`, `checkout`, and `getHolder` to be performed efficiently. In doing so, you must explain how you are using your chosen `java.util` class to orga-

nize your data and give the expected Big-Oh running time for each of these three operations in terms of  $n$ , the number of items in the library.

Your data representation (Circle one)	How data is organized
<code>ArrayList</code> <code>LinkedList</code> <code>HashMap</code> <code>TreeMap</code> <code>HashSet</code> <code>TreeSet</code>	

Operations	Expected Big-Oh efficiency
<code>add</code>	
<code>checkout</code>	
<code>getHolder</code>	

### Question 4 (Computer Science AB)

In approval voting there are at least two candidates and there are many voters. Each voter votes by submitting a ballot that represents a set of candidates of whom the voter approves. The winner of the election is the candidate that is listed on the most ballots, that is, the candidate who is approved by the most voters.

For example, suppose there are four candidates: Chris, Jamie, Pat, and Sandy. Seven voters vote as follows (order is unimportant, each ballot represents the set of candidates chosen by that voter).

Voter	Ballot
0	Chris, Jamie
1	Chris, Sandy
2	Chris, Sandy, Pat, Jamie
3	Pat
4	Sandy, Jamie
5	Sandy, Pat, Jamie
6	Jamie, Chris

The number of ballots on which each candidate was listed is as follows.

Candidate	Number of ballots
Chris	4
Jamie	5
Pat	3
Sandy	4

In this example, Jamie is the winner. If more than one candidate is tied with the most votes, then we consider the set of such candidates to be the winning set for this election. For example, if Chris appeared on voter 3's ballot along with Pat, then Chris would tie Jamie with five votes and the winning set would be {Chris, Jamie}.

We represent one voter's ballot as a `Set` that contains the names (as `Strings`) of the candidates approved by that voter. We represent the set of ballots for all voters in the election by the following partial class declaration.

```

public class VoterBallots
{
    private Map voteCount; // key is the candidate name, value is the
                          // number of votes received by that candidate

    // precondition: each entry in ballotList is a Set representing
    //                 one voter's ballot
    // postcondition: voteCount.get(candidate) is the total number of
    //                 times candidate appears on ballots in ballotList
    public VoterBallots(List ballotList)
    { /* to be implemented in part (a) */ }

    // postcondition: returns an Integer object with value equal to the
    //                 maximum number of votes associated with any
    //                 key in the Map voteCount
    private Integer maxVotes()
    { /* implementation not shown */ }

    // postcondition: returns a set containing the candidate(s)
    //                 with the most votes
    public Set candidatesWithMost()
    { /* to be implemented in part (b) */ }

    // other methods not shown
}

```

(a) Write the `VoterBallots` constructor. The constructor creates the `Map` `voteCount` and initializes it to hold one entry for each candidate that appears on at least one ballot in `ballotList`. The associated value for each candidate is the total number of ballots on which that candidate appears.

A solution that creates an unnecessary new instance of a `List` or `Set` will not receive full credit.

Complete the constructor below.

```

// precondition: each entry in ballotList is a Set representing
//                 one voter's ballot
// postcondition: voteCount.get(candidate) is the total number of
//                 times candidate appears on ballots in ballotList
public VoterBallots(List ballotList)

```

(b) Write the `VoterBallots` method `candidatesWithMost`. Method `candidatesWithMost` returns a `Set` of the name(s) of the candidate(s) in `voteCount` with the most votes.

In writing method `candidatesWithMost`, you may call the private helper method `maxVotes`.

Solutions that reimplement functionality provided by `maxVotes`, rather than invoking `maxVotes`, will not receive full credit. A solution that creates an unnecessary new instance of a `List` or `Set` will not receive full credit.

Complete method `candidatesWithMost` below.

```

// postcondition: returns a set containing the candidate(s)
//                 with the most votes
public Set candidatesWithMost()

```

(c) Assume that there are  $C$  candidates and  $V$  voters. What is the expected time complexity, in Big-Oh notation in terms of  $C$  and  $V$ , for your implementation of method `candidatesWithMost`?

## How to Get Involved

There are many ways college and university faculty members can help maintain the high standards of the AP Program:

- Participate in a college comparability study
- Be an AP Reader
- Contribute multiple-choice test items for the AP Exam
- Become an AP Faculty Consultant

For more information, please go to: [apcentral.collegeboard.com/highered/getinvolved](http://apcentral.collegeboard.com/highered/getinvolved)

## Contact Us

National Office  
Advanced Placement Program  
45 Columbus Avenue  
New York, NY 10023-6992  
212 713-8066  
E-mail: [ap@collegeboard.org](mailto:ap@collegeboard.org)

## 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<sup>®</sup>, the PSAT/NMSQT<sup>®</sup>, and the Advanced Placement Program<sup>®</sup> (AP<sup>®</sup>). 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. For further information, visit [www.collegeboard.com](http://www.collegeboard.com).