



AP[®] Computer Science AB 2003 Sample Student Responses

The materials included in these files are intended for 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 noncommercial, face-to-face teaching purposes. This permission does not apply to any third-party copyrights contained herein. This material may not be mass distributed, 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.

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 whose mission is to prepare, inspire, and connect students to college and opportunity. Founded in 1900, the association is composed of more than 4,300 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 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 equity and excellence, and that commitment is embodied in all of its programs, services, activities, and concerns.

For further information, visit www.collegeboard.com

Copyright © 2003 College Entrance Examination Board. All rights reserved. College Board, Advanced Placement Program, AP, AP Vertical Teams, APCD, Pacesetter, Pre-AP, SAT, Student Search Service, and the acorn logo are registered trademarks of the College Entrance Examination Board.

AP Central is a trademark owned by the College Entrance Examination Board. PSAT/NMSQT is a registered trademark jointly owned by the College Entrance Examination Board and the National Merit Scholarship Corporation. Educational Testing Service and ETS are registered trademarks of Educational Testing Service. Other products and services may be trademarks of their respective owners.

For the College Board's online home for AP professionals, visit AP Central at apcentral.collegeboard.com.

```
class StringTokenizer
{
    public:
        StringTokenizer(const apstring & str);
        int NumTokens() const; // myTokens.length()
        apstring KthToken(int k) const; // myTokens[k]
```

```
private:
    - vector<apstring> myTokens;
```

```
};
```

Part (b) begins on page 22.

GO ON TO THE NEXT PAGE.

- (b) Write free function `CreateAcronym`, which is described as follows. `CreateAcronym` takes a string, `str`, as a parameter and returns a string that is the acronym formed from the first character of each token of `str`. The following table shows several examples of calls to `CreateAcronym`.

<u>str</u>	String returned by <u>CreateAcronym(str)</u>
"red orange yellow green blue indigo violet"	"roygbiv"
" as soon as possible"	"asap"
"Rolling on the floor laughing"	"Rotfl"
"As Far As I Know"	"AFAIK"

More formally, the acronym for a string `str` is formed by concatenating the first character of each token of `str` in the same order that the tokens appear in `str`.

In writing `CreateAcronym`, you must use the `StringTokenizer` class you designed in part (a). To receive full credit, the tokens of `str` must only be obtained by using the member functions of the `StringTokenizer` class that implement the specification given at the beginning of the question. Assume that the `StringTokenizer` has been implemented as specified.

Complete function `CreateAcronym` below.

```
apstring CreateAcronym(const apstring & str)
```

```
{
    apstring acro, token;
    StringTokenizer tokenizer(str);
    for (int k=0; k < tokenizer.NumTokens(); ++k)
    {
        token = tokenizer.KthToken(k);
        acro += token[0];
    }
    return acro;
}
```

GO ON TO THE NEXT PAGE.

```
class StringTokenizer
{
    public:

        StringTokenizer();
        StringTokenizer (apstring sourceString);

        int NumTokens() const;
        apstring GetToken (int index) const;
```

```
private:
    apvector <apstring> TokenList;
    int NumTokens;
    apstring OriginalString;
```

```
};
```

Part (b) begins on page 22.

GO ON TO THE NEXT PAGE.

- (b) Write free function `CreateAcronym`, which is described as follows. `CreateAcronym` takes a string, `str`, as a parameter and returns a string that is the acronym formed from the first character of each token of `str`. The following table shows several examples of calls to `CreateAcronym`.

<u>str</u>	String returned by <u>CreateAcronym(str)</u>
"red orange yellow green blue indigo violet"	"roygbiv"
" as soon as possible"	"asap"
"Rolling on the floor laughing"	"Rotfl"
"As Far As I Know"	"AFAIK"

More formally, the acronym for a string `str` is formed by concatenating the first character of each token of `str` in the same order that the tokens appear in `str`.

In writing `CreateAcronym`, you must use the `StringTokenizer` class you designed in part (a). To receive full credit, the tokens of `str` must only be obtained by using the member functions of the `StringTokenizer` class that implement the specification given at the beginning of the question. Assume that the `StringTokenizer` has been implemented as specified.

Complete function `CreateAcronym` below.

```
apstring CreateAcronym(const apstring & str)
{
    StringTokenizer TList(str);
    apstring Acronym=" ";
    for(int i=0; i < TList.NumTokens(); i++)
    {
        Acronym += TList.GetToken(i).substr(0,1);
    }
    return (Acronym);
}
```

GO ON TO THE NEXT PAGE.

AB4 C

```
class StringTokenizer
```

```
{
```

```
    public:
```

```
        String tokenizer (apstring & s);
```

```
        int NumTokens (); const;
```

```
        apstring KthToken (int k);
```

```
    private:
```

```
        apstring string;
```

```
};
```

Part (b) begins on page 22.

GO ON TO THE NEXT PAGE.

- (b) Write free function `CreateAcronym`, which is described as follows. `CreateAcronym` takes a string, `str`, as a parameter and returns a string that is the acronym formed from the first character of each token of `str`. The following table shows several examples of calls to `CreateAcronym`.

<u>str</u>	String returned by <u>CreateAcronym(str)</u>
"red orange yellow green blue indigo violet"	"roygbiv"
" as soon as possible"	"asap"
"Rolling on the floor laughing"	"Rotfl"
"As Far As I Know"	"AFAIK"

More formally, the acronym for a string `str` is formed by concatenating the first character of each token of `str` in the same order that the tokens appear in `str`.

In writing `CreateAcronym`, you must use the `StringTokenizer` class you designed in part (a). To receive full credit, the tokens of `str` must only be obtained by using the member functions of the `StringTokenizer` class that implement the specification given at the beginning of the question. Assume that the `StringTokenizer` has been implemented as specified.

Complete function `CreateAcronym` below.

```

apstring CreateAcronym(const apstring & str)
{
    apstring Acronym = "";
    StringTokenizer (apstring str) S;
    int n = S.NumTokens();
    int a;
    for (a = 1; a <= n; a++)
    {
        Acronym += S.KthToken(a)[0];
    }
    return Acronym;
}

```

GO ON TO THE NEXT PAGE.