

3OBPABM in C++
1992 Advanced Placement Exam (AB)

Owen Astrachan

PROBLEM 1:

As is, doesn't change

PROBLEM 2:

As is, doesn't change

PROBLEM 3:

As is, doesn't change

PROBLEM 4:

Assume that A is an array of n positive integers and that the following assertion is true.

$$A[0] > A[k] \text{ for all } k \text{ such that } 0 < k < n$$

Which of the following is a valid conclusion?

- A. The array is sorted in ascending order.
- B. The array is sorted in descending order.
- C. All values in the array are identical.
- D. $A[0]$ holds the smallest value in the array.
- E. $A[0]$ holds the largest value in the array

(only difference is starting with index 0 instead of index 1)

PROBLEM 5:

A program to print a calendar includes the following code.

```
for(month = 1; month <= 12; month++)
{
    PrintHeading(month, year);
    PrintDays(month, year);
}
```

The `PrintDays` function includes the following code.

```
PrintSpaces(month, year);
for(day=1; day <= NumDaysIn(month, year); day++)
{
    cout << setw(3) << day;
    if (EndOfWeek(day, month, year))
        cout << endl;
}
```

If, when the program is run, every week on the calendar printed has eight days, which of the following functions is most likely to contain the bug?

(choices do not change)

- A. `PrintHeading`
- B. `PrintDays`
- C. `PrintSpaces`
- D. `NumDaysIn`
- E. `EndOfWeek`

PROBLEM 6:

The following code is designed to set *index* to the location of the first occurrence of *goal* in the array *A*, and to set *index* to -1 if *goal* does not occur in *A*.

```
index = 0;
while (A[index] != goal)
{
    index++;
}
if (A[index] != goal)
{
    index = -1;
}
```

Which of the following describe the condition under which this program segment will fail to perform the task described?

(choices do not change)

- A. Whenever *goal* is the first element of the array
- B. Whenever *goal* is the last element of the array
- C. Whenever *goal* is not present in the array
- D. Whenever *goal* is 0
- E. Whenever $goal = A[goal]$

PROBLEM 7:

A C++ compiler runs on several different types of computers, ranging from microcomputers to mainframes. For this compiler, which of the following might be different on the different machines?

- I. The value of `INT_MAX` in `<limits.h>`
 - II. The number of reserved words
 - III. The largest value represented by the type `double`
- A. I only
 - B. III only
 - C. I and II
 - D. I and III
 - E. II and III

PROBLEM 8:

Consider the task of moving m items from the rear to the front of an n -item list. For example, the movement of 3 items from the rear to the front in the 8-item list

(12, 32, 22, 44, 55, 21, 77, 34)

creates the list

(21, 77, 34), 12, 32, 22, 44, 55)

The following algorithm performs this task (lists are indexed beginning with position 0).

```

for(k=0;k < m;k++){
    save the item that is at the end of the list;
    shift items in positions 0..n-2 to positions 1..n-1
    put the saved item into position 0
}

```

If the list is stored in an array of n elements, which of the following best describes the running time of this algorithm in terms of the number of shifts of individual items?

- A. m^2
- B. n^2
- C. mn
- D. m
- E. n

(only change is 0-index rather than 1 index)

Questions 9-10 refer to the following function.

```

int Answer(int n)
{
    if (n == 1)
    {
        return 1;
    }
}

```

```
    else
    {
        return 2 * answer(n-1);
    }
}
```

PROBLEM 9:

(question and choices do not change)

What is the value of the expression `answer(5)`?

- A. 2
- B. 8
- C. 10
- D. 32
- E. 120

PROBLEM 10:

(question and choices do not change)

If n is a positive integer, how many times will `Answer` be called to evaluate `Answer(n)` (including the initial call)?

- A. 2
- B. n
- C. $2n$
- D. n^2
- E. 2^n

Questions 11-12 are based on the following code.

```
bool LookUp(const Dictionary & d; int target)
// postcondition: returns true if the value of target
//                is in dictionary d; otherwise returns false
{
    if (d == NULL)
        return false;
    else if (d->value == target)
        return true;
    else if (d->value > target)
        return LookUp(d->left, target);
    else
        return LookUp(d->right, target);
}
```

PROBLEM 11:

(question and choices do not change)

Function `LookUp` is best characterized as performing a search in

- A. an ordered binary (search) tree
- B. an unordered binary tree
- C. an ordered linear linked list
- D. an unordered linear linked list
- E. a hash table

PROBLEM 12:

If function `LookUp` comiles without errors, then the type definitions involved could be chosen from which of the following?

- I.

```
struct Item
{
    int value;
    Item * left;
    Item * right;
};

typedef Item * Dictionary;
```
- II.

```
struct Item
{
    int value;
    Item * left;
    Item * right;
};

typedef Item Dictionary;
```
- III.

```
struct Item
{
    int value;
    Item left;
    Item right;
};

typedef Item Dictionary;
```

- A. I only
- B. II only
- C. III only
- D. I and III only
- E. I, II, and III

PROBLEM 13:

A program includes the following declarations.

```
struct Node
{
    ValueType value;
    Node * left;
    Node * right;
};
```

A function of the program includes the following code segment.

```
tree->left = tree->right;
tree->right = tree;
```

Which of the following must be true of this segment? (Assume that `tree != NULL` and that all fields of `tree` have been initialized.)

(choices do not change)

- A. It will trigger a compile-time error message.
- B. It will trigger a run-time error message.
- C. It will result in an infinite loop.
- D. It will result in an infinite recursion.
- E. None of the above.

PROBLEM 14:

Consider the following definitions and declarations

```
struct TreeNode
{
    int info;
    TreeNode * left;
    TreeNode * right;
};

int m(int a, int b)
{
    if (a >= b) return a; else return b;
}

int f(TreeNode *t)
{
    if (t->left == 0 && t->right == 0) return t->info;
    else if (t->left == 0) return m(t->info, f(t->right));
    else if (t->right == 0) return m(t->info, f(t->left));
    else return m( m(t->info, f(t->left)), f(t->right));
}
```

(questions and choices do not change) If, on entrance to `f`, `t` is a non-empty tree, then `f` evaluates to which of the following?

- A. the greatest value of *info* stored in the tree
- B. the least value of *info* stored in the tree
- C. the value of *info* at the root of the tree
- D. the value of *info* at the last node of the tree visited during execution of *f*
- E. the first duplicated value of *info* encountered during execution of *f*

PROBLEM 15:

Consider function `Print` below.

```

struct Node
{
    char info;
    Node * next;
    Node(char ch, Node * link)
    {
        info = ch;
        next = link;
    }
};

void Print()
{
    Node * list;
    Node * temp;
    char letter;

    list = NULL;
    for(letter='1'; letter <= '5'; letter++)
    {
        temp = new Node(letter,list);
        list = temp;
    }

    while (temp != NULL)
    {
        cout << temp->info;
        temp = temp->next;
    }
    cout << endl;
}

```

(question and choices do not change)

What will be printed as a result of calling function `Print`?

- A. 54321
- B. 12345
- C. 51
- D. 5
- E. 1

Questions 16-17 refer to the following Boolean expression.

```
(i <= n && a[i] == 0) || (i >= n && a[i-1] == 0)
```

PROBLEM 16:

Under which of the following conditions must the Boolean expression have the value `true`?

- A. `1 <= n || i >= n`
- B. `a[i] == 0 && a[i-1] == 0`
- C. `i == n`
- D. `i < n`
- E. `i > n`

PROBLEM 17:

(question and choices do not change)

Evaluation of the Boolean expression is guaranteed to cause a run-time error under which of the following conditions?

- A. `i < 0`
- B. Neither `a[i]` nor `a[i-1]` has the value zero.
- C. Array `a` is of size `n`.
- D. Array `a` is of size 2.
- E. None of the above.

PROBLEM 18:

Consider the following declarations and definitions.

```
struct List
{
    int items[MaxLength];
    int numItems;
};

void Find(const List & list, int num,
          bool & found, int & loc)
// precondition: 0 <= list.numItems < MaxLength
{
    found = false;
    for(loc=0; loc < list.numItems; loc++)
    {
        if (list.items[loc] == num)
        {
            found = true;
        }
    }
}
```



```

        break;
    }
}

```

Which of the following is a correct postcondition for function `Find`?

- A. `found`
- B. `found && loc >= list.numItems`
- C. `list.item[loc] == num) || loc == list.numItems`
- D. `loc == list.numItems`
- E. `!found && loc < list.numItems`

Questions 19-20 are based on the following information.

As is, replace the word “record” with “struct”.

PROBLEM 19:

As is, no changes

PROBLEM 20:

As is, replace “standard Pascal” with C++.

PROBLEM 21:

The Boolean expression

```
num > max || !(max < num)
```

can be simplified to

- A. `max != num`
- B. `max == num`
- C. `num < max && !(max < num)`
- D. `false`
- E. `true`

PROBLEM 22:

As is

PROBLEM 23:

As is

PROBLEM 24:

Suppose that a queue of integers has been implemented supporting the usual queue operations *Enqueue*, *Dequeue*, and *IsEmpty*. Consider the following pseudocode (`q` is a queue).

```

int time,value;
do
{
    time = 0;
    q.Dequeue(value);
    do
    {
        value--;
        time++;
    } while (value != 0 && time < limit);

    if (value > 0) q.Enqueue(value);
} while (! q.IsEmpty());

```

(question and choices do not change)

Suppose that initially the values in the queue are:

1, 10, 8, 5, 12

(1 is at the front of the queue, 12 is at the end of the queue.)

Which of the following is the least value of `limit` that would ensure that the total number of Dequeue operations is 6 or less?

- A. 3
- B. 5
- C. 6
- D. 7
- E. 10

Questions 25-26 concern the definition of boolean operators **cand** and **cor** which aren't useful in C++ since these operators are short-circuited by the language definition. Questions 25 and 26 “do not port” to C++.

PROBLEM 25:

No C++ analog

PROBLEM 26:

No C++ analog

PROBLEM 27:

Assume that the following definitions have been made.

```

const int MaxNum = <some positive integer>;
bool list [MaxNum+3];

```

Consider the following code segment.

```

for(i=2; i <= MaxNum; i++)
{
    for(j=1; j <= MaxNum/i; j++)
    {
        list[i*j] = ! list[i*j];
    }
}

```

(question and choices do not change)

For i in the range $2..MaxNum$, which of the following characterizes the entries of `list` that will have value `true` after the segment above has executed?

- A. `list[i] = true` for no values of i .
- B. `list[i] = true` for all values of i .
- C. `list[i] = true` for all values of i that are even.
- D. `list[i] = true` for all values of i that are prime.
- E. `list[i] = true` for all values of i that are perfect squares.

Questions 28-29 are based on the following information.

Assume that variables of type `LongInt` can represent integers of up to fifty digits. Also assume that the following operators are to be written so as to operate on `LongInt` variables, and that these are the only operators written.

```

istream & operator >> (istream & input, LongInt & long)
// reads a long integer from the stream input

ostream & operator << (ostream & output, const LongInt & long)
// writes a long integer to the stream output

LongInt operator +(const LongInt & a, const LongInt & b)
// returns a + b

bool operator < (const LongInt & a, const LongInt & b)
// returns true if a < b, otherwise returns false

LongInt & operator = (int num)
// assignment operator, assign int value num to LongInt

```

PROBLEM 28:

Of the following pairs of operators, which should be coded and tested first in order to facilitate the debugging of the other operators? (Assume the runtime system provides no debugging facilities.)

- A. `operator >>` and `operator +`
- B. `operator <<` and `operator +`
- C. `operator <` and `operator +`
- D. `operator =` and `operator <<`
- E. `operator >>` and `operator =`

PROBLEM 29:

In order to simulate the loop

```
for(i=1; i < n; i++)
    cout << i;
```

the following code is used (where `i`, `one`, and `n` are `LongInt` variables).

```
one = 1;
for(i=1; i < n; <statement>)
    cout << i;
```

Which of the following should take the place of `<statement>` in order to simulate the loop correctly?

- A. `n = one + i`
- B. `one = i + n`
- C. `one = i + i`
- D. `i = one + n`
- E. `i = i + one`

PROBLEM 30:

Suppose a binary tree is defined as follows:

```
struct Tree
{
    Tree * left;
    Tree * right;
};
```

Consider the following function:

```
int Doit(Tree * t)
{
    if (t == NULL)
        return 0;
    else
        return Max(Height(t->left) + Height(t->right),
                   Doit(t->left),Doit(t->right));
}
```

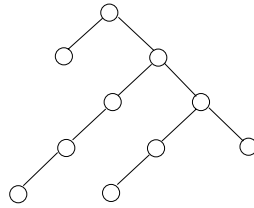
(question and choices do not change)

Suppose that the function `Max` returns the largest of its three integer arguments, and function `Height` returns the height of its tree argument, where the height of a tree is defined as follows.

The height of an empty tree is 0.

The height of a nonempty tree is the number of nodes on the longest path from the root to a leaf of the tree.

What value is returned when `Doit` is passed an argument representing the following tree of height 5?



- A. 4
- B. 5
- C. 6
- D. 8
- E. 10

Questions 31-32 are based on the following code framework.

```
int i = 1;
int v = 1;
int n;
cin >> n;
while <condition>
{
    <body>
}
cout << v << endl;
```

The placeholders `<condition>` and `<body>` are to be replaced with code so that whenever the value read into variable n is positive, the value output is $n!$ (n factorial). Further the expression $v = i!$ is to be maintained as an invariant of the while loop.

PROBLEM 31:

Which of the following choices for `<body>` maintains $v = i!$ as the loop invariant?

- A. `i += 1;` `v *= i;`
- B. `v *= i;` `i += 1;`
- C. `i += 1;` `v = n * i;`
- D. `v = n * i;` `i += 1;`
- E. `i = i * (i-1);` `v += 1;`

PROBLEM 32:

Assume that `<body>` has been replaced with code that maintains $v = i!$ as the loop invariant. Which of the following choices for `<condition>` ensures that if the loop terminates, the value $n!$ is output?

- A. `i == n`

- B. $i \neq n$
- C. $i == v$
- D. $i \neq v$
- E. $i == v * n$

PROBLEM 33:

Consider the type and function definitions below

```

struct Node
{
    int data;
    Node * next;
};

Node * Mystery(Node * list, Node * soFar)
{
    if (list == NULL)
        return soFar;
    else
    {
        Node * temp = list->next;
        list->next = soFar;
        return Mystery(temp,list);
    }
}

```

(question and choices do not change)

Suppose that p points the list (1,2,3,4). What is the list returned by `Mystery(p,NULL)`?

- A. NULL
- B. (1)
- C. (1,1,1,1)
- D. (4,3,2,1)
- E. (1,2,3,4)

Questions 34-36 are based on the following pseudocode for a function P that copies items from an array A containing n distinct items into a binary search tree T and then prints the items.

Function P

Step 1:	Initilize binary search tree T to be empty.
Step 2:	for ($i=0$; $i < n$; $i++$) Use the standard algorithm for insertion into a binary search tree to inset item $A[i]$ into T .
Step 3:	Print the items stored in T , using an inorder traversal.

Assume that the inset operation used in Step 2 does no balancing of T .

PROBLEM 34:

Which fo the of the following best characterize the output produced in step 3 of function P .

- A. The items are printed in the same order in which they appear in array A .
- B. The items are printed in sorted order.
- C. The items are printed in random order.
- D. The items are printed in the reverse of the order in which they appear in array A .
- E. The value stored in $A[0]$ is printed n times

PROBLEM 35:

As is, no changes

PROBLEM 36:

As is, no changes

Questions 37-38 refer to the following information.

Consider two implementations for a data structure to represent a pair of strings up to 20 characters each. In both cases the type of the data structure is `StringPair`.

Implementation A

```

struct String
{
    char chars[20];
    int length;
};

struct StringPair
{
    String s1, s2;
};

```

Implementation B

```

struct String
{
    int first,last;
};

struct StringPair
{
    char storage[40];
    String s1,s2;
};

```

In Implementation A, `StringPair` is a struct that contains two single strings. A single string is represented by a sequence of characters stored in an array (starting in position 0) and an integer representing the length of the sequence.

In Implementation B, `StringPair` is a struct with three fields. The field called `storage` is an array of characters; the fields `s1` and `s2` each contain a pair of integers, which are indexes into the storage array. A single string consists of the sequence of characters `storage[first..last]`.

PROBLEM 37:

As is, doesn't change

PROBLEM 38:

add an array cell with index 0 to the front of the diagram and remove array cell with index 40. Otherwise doesn't change.

Questions 39-40 are based on the following information.

A function T is defined as follows.

$$\begin{aligned} T(0) &= 1 \\ T(1) &= 3 \\ T(N) &= 2 * T(N - 1) - T(N - 2) \text{ for } N > 1 \end{aligned}$$

Consider the following two functions, `RecurT` and `IterateT`, for computing $T(N)$.

```
int RecurT(int N)
{
    if (0 == N)        return 1;
    else if (1 == N)   return 3;
    else return 2*RecurT(N-1) - RecurT(N-2);
}
```

```
int IterateT(int N)
{
    int temp1, temp2, temp3, k;
    if (0 == N)        return 1;
    else if (1 == N)   return 3;
    else
    {
        temp1 = 1;
        temp2 = 3;
        for(k=2; k <= N; k++)
        {
            temp3 = 2 * temp2 - temp1;
            temp1 = temp2;
            temp2 = temp3;
        }
        return temp3;
    }
}
```

PROBLEM 39:

As is, doesn't change

PROBLEM 40:

Change `div` to `/`, otherwise doesn't change