

APS105 ConcepTests

Dan Zingaro

April 2, 2010

Introduction

Concept Test

Consider the following statements about compilers: (I) Compilers translate C code to machine language

(II) Compilers must know the type of machine on which the generated machine code will run

(III) Compilers detect all possible program errors

(IV) A compiler is not required if you write directly in machine language

Which of these statements are true?

- ▶ A. All statements are true
- ▶ B. (I) and (III) only
- ▶ C. (I), (II), and (IV) only
- ▶ D. (I), (II), and (III) only

ConceptTest

What is the output of the following printf function call?

```
printf ("I said, \"hi\");
```

- ▶ A. I said, "hi"
- ▶ B. I said, "hi
- ▶ C. I said, "hi""
- ▶ D. I said, "hi\"
- ▶ E. Compilation error

Concept Test

How many different states can we represent using five bits?

- ▶ A. 10
- ▶ B. 16
- ▶ C. 25
- ▶ D. 32

ConceptTest

After we say `int age;`, we can refer to the selected memory location with the identifier `age`. Which integer might already be stored in that memory location?

- ▶ A. 0
- ▶ B. 1
- ▶ C. -1
- ▶ D. All of the above are possible

ConcepTest

What is the value of y after the execution of this code?

```
int x = 37;  
int y = x + 2;  
x = 20;
```

- ▶ A. 39
- ▶ B. 2
- ▶ C. 22
- ▶ D. 20

ConceptTest

What is the result of the following code?

```
int age;  
age = 0;  
scanf ("%d", age);
```

- ▶ A. Ask the user for input, and *properly* store it in the memory location for age
- ▶ B. Ask the user for input, and store it in memory location 0
- ▶ C. Ask the user for input, and store it in an arbitrary memory location

Expressions and Operators

ConceptTest

What is the result of the following expression?

$$5 / 4 + (5.0 / 2)$$

- ▶ A. 3.75
- ▶ B. 3.5
- ▶ C. 5/6
- ▶ D. 5/6.5

ConceptTest

What does this program fragment print?

```
int i = 3;
  int j = i++;
  int k = ++i;
  printf ("%d %d\n", j, k);
```

- ▶ A. 3 3
- ▶ B. 3 4
- ▶ C. 3 5
- ▶ D. 4 4
- ▶ E. 4 5

ConceptTest

What is the value of variable `i` after the following code runs?

```
int i;  
i = 'P' - 'D';
```

- ▶ A. Error!
- ▶ B. 12
- ▶ C. 13
- ▶ D. 14

Decision-Making

ConceptTest

What is the output of the following code?

```
int x = 2 < 3;  
int y = 2 != 2;  
int z = 2 == 2 == 2;  
printf ("%d\n", x + y + z);
```

- ▶ A. 0
- ▶ B. 1
- ▶ C. 2
- ▶ D. 3
- ▶ E. Compilation error!

ConceptTest

Here are some if-statements, with placeholders for their bodies.

(I) `if (2 < 3) S1;`

(II) `if (2) S2;`

(III) `if (0 == 0) S3;`

(IV) `if (0) S4;`

Which statements will be executed?

- ▶ A. (I)
- ▶ B. (I) and (II)
- ▶ C. (I) and (III)
- ▶ D. (I), (II) and (III)
- ▶ E. (II) and (IV)

ConceptTest

Consider the following C statement.

```
printf ("hi") || printf ("bye");
```

What is the output of running this statement? (As return value, `printf` returns the number of characters printed.)

- ▶ A. No output
- ▶ B. hi
- ▶ C. hibye
- ▶ D. bye

ConceptTest

What is the output of the following code, assuming that `x` is an `int` variable with value 4?

```
if (x <= 2) {  
  if (x == 4)  
    printf ("one\n");  
}  
else  
  printf ("two\n");
```

- ▶ A. one
- ▶ B. two
- ▶ C. one two
- ▶ D. No output

ConcepTest

For which integer values of x will the following code print `hi`?

```
switch (x - 3) {  
    case 7: break;  
    case 6:  
    case 4: printf ("hi\n");  
}
```

- ▶ A. 9 7
- ▶ B. 6 4
- ▶ C. 4
- ▶ D. 7

Repetition

ConceptTest

What is the value of `x` after this code executes?

```
int x = 4;
do {
    x -= 5;
    x++;
} while (x >= 0);
```

- ▶ A. -1
- ▶ B. -2
- ▶ C. -3
- ▶ D. -4
- ▶ E. 0

ConceptTest

How many times does the following for-loop execute?

```
for (int i = 2; i >= -9; i -= 3)
    printf ("%d\n", i);
```

- ▶ A. 2
- ▶ B. 3
- ▶ C. 4
- ▶ D. 5
- ▶ E. Infinite

ConcepTest

How many times does the following for-loop execute?

```
int i = 2;
for (; i >= -9;)
    printf ("%d\n", i);
```

- ▶ A. 2
- ▶ B. 3
- ▶ C. 4
- ▶ D. 5
- ▶ E. Infinite

Functions

ConcepTest

What is printed by the following program fragment?

```
void sayHi (void) {  
    printf ("Hi! ");  
    sayBye();  
}
```

```
void sayBye (void) {  
    printf ("Bye! ");  
}
```

```
int main (void) {  
    sayHi();  
    sayHi();  
    ...  
}
```

- | | | |
|----------------------|--|------------------------------|
| A. Hi! Bye! Hi! Bye! | | B. Hi! Hi! |
| C. Hi! Bye! | | D. Hi! Bye! infinitely often |

ConceptTest

How would we call the following function to print 10 x's?

```
void printRow (char c, int n) {  
    for (int i = 1; i <= n; i++)  
        printf ("%c", c);  
    printf ("\n");  
}
```

- ▶ A. `printRow (10, x);`
- ▶ B. `printRow (x, 10);`
- ▶ C. `printRow (10, 'x');`
- ▶ D. `printRow ('x', 10);`

ConceptTest

Consider the following function.

```
double f (double x) {  
    double y;  
    if (x < 0)  
        y = x * x * x;  
    else  
        y = x * x;  
    return y;  
}
```

Assuming the declaration of double variable q , what is assigned to q by the statement $q = f(1) + 2 + f(2)$?

- ▶ A. 6
- ▶ B. 7
- ▶ C. 9
- ▶ D. 11

ConceptTest

What is printed by the following code?

```
void addOne (int *i, int *j) {  
    *i = *i + 1;  
    *j = *j + 1;  
}
```

```
int main (void) {  
    int a = 3;  
    addOne (&a, &a);  
    printf ("%d\n", a);  
    return 0;  
}
```

- ▶ A. 3
- ▶ B. 4
- ▶ C. 5
- ▶ D. Error!

ConceptTest

Which variables can be accessed at the line with the comment?

```
void sample(void) {  
    int i = 3;  
    if (i == 3) {  
        int j;  
    }  
    {  
        int i;  
    }  
    int k;  
    //What can I access here?  
}
```

- ▶ A. i
- ▶ B. i, j
- ▶ C. i, j, k
- ▶ D. i, k
- ▶ E. k

ConcepTest

What is printed by this program fragment?

```
...
int doSomething (int x) {
    static int y = 1;
    y += x;
    return y;
}

int main (void) {
    doSomething (3);
    doSomething (4);
    printf ("%d\n", doSomething (1));
    ...
}
```

A. 7 | B. 8 | C. 9 | D. 1 | E. 2

ConceptTest

```
int player;
```

Assume `player` has value 1 or 2; if it has value 1, we want to change it to 2, and if it has value 2, we want to change it to 1. Which of the following code fragments will do this?

- ▶ A. `player = !player;`
- ▶ B. `player = player % 2;`
- ▶ C.

```
if (player == 1)
    player = 2;
if (player == 2)
    player = 1;
```
- ▶ D. More than one of the above
- ▶ E. None of the above

ConceptTest

```
char ch;
```

Which of the following code fragments could be used to ensure that we get a y or n response?

- ▶ A.
do {
 ...
} while (ch != 'y' && ch != 'n');
- ▶ B.
do {
 ...
} while (ch != 'y' || ch != 'n');
- ▶ C.
do {
 ...
} while (!(ch == 'y' || ch == 'n'));
- ▶ D. More than one of the above
- ▶ E. None of the above

ConceptTest

In which order will the following fragments result in a program that asks for five **positive** integers and prints their maximum?

- (1) `scanf ("%d", &num);`
- (2) `if (num > max)`
- (3) `printf ("%d\n", max);`
- (4) `max = num;`
- (5) `int num;`
- (6) `}`
- (7) `int max = 0;`
- (8) `for (int i = 1; i <= 5; i++) {`

- ▶ A. (5), (7), (1), (2), (4), (8), (1), (2), (4), (6), (3)
- ▶ B. (5), (7), (8), (1), (2), (4), (6), (3)
- ▶ C. (5), (7), (1), (2), (8), (1), (2), (4), (6), (3)
- ▶ D. (8), (5), (7), (1), (2), (4), (6), (3)
- ▶ E. None of the above

Arrays

ConceptTest

What is the value of `a[4]` after the following code runs?

```
int a[5];  
a[1] = 5;  
a[2] = 4;  
a[4] = a[2] + 1;
```

- ▶ A. 5
- ▶ B. 9
- ▶ C. 4
- ▶ D. 1

ConceptTest

Here is the outline of a program that reads N numbers from the user. What code should replace //Loop if we want to print those numbers in reverse?

```
int a[N], i;
for (i = 0; i < N; i++)
    scanf("%d", &a[i]);
//Loop
```

- ▶ A.
for (i = N; i >= 0; i--)
 printf(" %d", a[i]);
- ▶ B.
for (i = N - 1; i >= 0; i--)
 printf(" %d", a[i]);
- ▶ C.
for (i = N; i > 0; i--)
 printf(" %d", a[i-1]);
- ▶ D. All of the above
- ▶ E. B and C only

ConceptTest

Which of the following can be used to initialize all five elements of array a to 0?

- ▶ A. `int a[0];`
- ▶ B. `int a[] = {0, 0, 0, 0, 0};`
- ▶ C. `int a[] = {0};`
- ▶ D. All of the above
- ▶ E. B and C only

ConceptTest

Choose the array that has the largest maximum segment sum.

- ▶ A. [4, -3, 9, -5]
- ▶ B. [-5, 9, -6, 2]
- ▶ C. [-8, -2, -1] (careful!)

ConceptTest

What is the proper loop structure for accessing the elements in the following table **by column**, from top to bottom? We want to access the elements in the first column, then those in the second column, etc.

```
int marks[3][4];
```

▶ A.

```
for (int i = 0; i < 4; i++)  
    for (int j = 0; j < 3; j++)  
        ... access marks[i][j]
```

▶ B.

```
for (int i = 0; i < 3; i++)  
    for (int j = 0; j < 4; j++)  
        ... access marks[i][j]
```

▶ C.

```
for (int i = 0; i < 4; i++)  
    for (int j = 0; j < 3; j++)  
        ... access marks[j][i]
```

ConceptTest

What are the value of array a's elements after the following code?

```
int a[2] = {2, 4};  
int *p = &a[1];  
*p = 1;  
p--;  
*(p+1) = 6;
```

- ▶ A. [2, 4]
- ▶ B. [1, 6]
- ▶ C. [6, 1]
- ▶ D. [2, 6]
- ▶ E. [6, 4]

ConceptTest

```
int a[2] = {4, 2};  
int *p = &a[1];
```

Which of the following is true after the code runs?

- ▶ A. $p < \&a[0]$
- ▶ B. $p == \&a[0]$
- ▶ C. $p > \&a[0]$

ConceptTest

What does the following code fragment do?

```
void storeZeros (int n, int a[n]) {  
    int i;  
    for (i = 0; i < n; i++)  
        a[i] = 0;  
}
```

...

```
int a[10];  
storeZeros (5, &a[5]);
```

- ▶ A. Stores zeros in a[0], a[1], ..., a[4]
- ▶ B. Stores zeros in a[0], a[1], ..., a[5]
- ▶ C. Stores zeros in a[5], a[6], ..., a[9]
- ▶ D. Erroneously tries to store zeros in a[5], a[6], ..., a[10]
- ▶ E. Causes a compiler error

Strings

ConceptTest

Fill in the comment with the correct code.

```
char s[50];  
char *t = "abc";  
//make s the string "abc"
```

▶ A.

```
s = "abc";
```

▶ B.

```
s = t;
```

▶ C.

```
s[0] = t[0];
```

```
s[1] = t[1];
```

```
s[2] = t[2];
```

▶ D.

```
s[0] = t[0];
```

```
s[1] = t[1];
```

```
s[2] = t[2];
```

```
s[3] = t[3];
```

▶ E. A or B

ConceptTest

Assume that the user types `abc defg`, but that only `abc` is stored in the string `s`. Which of the following could have been used to read the string?

- ▶ A. `scanf ("%s", s);`
- ▶ B. `gets (s);`
- ▶ C. `fgets (s, 3, stdin);`

ConceptTest

What is the result of this code?

```
char s[] = "hi\0hey";
```

- ▶ A. The three characters 'h', 'i', '\0' are stored in s;
strlen (s) == 2
- ▶ B. The seven characters
'h', 'i', '\0', 'h', 'e', 'y', '\0' are stored in s;
strlen (s) == 2
- ▶ C. The seven characters
'h', 'i', '\0', 'h', 'e', 'y', '\0' are stored in s;
strlen (s) == 6
- ▶ D. Error!

ConceptTest

Here is an implementation of `strcmp`, with the while-condition removed. Which while-condition makes the code correct?

```
int my_strcmp (const char *s1, const char *s2) {
    while (...) {
        s1++;
        s2++;
    }
    if (*s1 == '\0' && *s2 == '\0') return 0;
    else if (*s1 == '\0') return -1;
    else if (*s2 == '\0') return 1;
    else return *s1 - *s2;
}
```

- ▶ A. `*s1 == *s2`
- ▶ B. `*s1 == *s2 && *s1 != '\0' && *s2 != '\0'`
- ▶ C. `*s1 == *s2 || *s1 != '\0' || *s2 != '\0'`
- ▶ D. A and B
- ▶ E. B and C

ConceptTest

What is stored in `s` after the following code executes?

```
char s[10] = "one";  
char t[10] = "two";  
strcpy (s, t);  
strcpy (t, s);  
strcat (s, t);
```

- ▶ A. one one
- ▶ B. two two
- ▶ C. one two two one two
- ▶ D. two
- ▶ E. one

ConceptTest

Assume string `s2` exists in string `s1`. We want to know the index of `s1` at which `s2` starts. For example, if `s1` is `school` and `s2` is `ool`, we want the value 3. Which of the following lines of code does this?

- ▶ A. `strstr (s1, s2) - s2`
- ▶ B. `strstr (s1, s2) - s1`
- ▶ C. `strstr (s1, s2) - s2 + 1`
- ▶ D. `strstr (s1, s2) - s1 + 1`

ConceptTest

Assume we want to store the strings `Canada`, `Ontario`, and `flag`. Which of the following declarations could be used?

- ▶ A. `char words[3][7];`
- ▶ B. `char words[3][8];`
- ▶ C. `char words[3][9];`
- ▶ D. All of the above
- ▶ E. None of the above

ConceptTest

Consider the following function to copy string q into string p.

```
void stringCopy (char *p, char *q) {  
    while ((*p = *q) != \0) {  
        p++;  
        q++;  
    }  
}
```

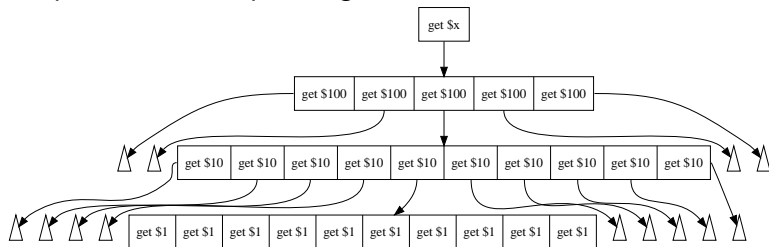
Does the code properly put a '\0' at the end of p?

- ▶ A. Yes, because the '\0' is copied just before the while-condition fails
- ▶ B. No, because there is no *p = \0; after the loop
- ▶ C. No, because no characters at all are copied into p

Recursion

Concept Test

What is the value of x in this tree? That is, how much money is the person at the top raising?



- ▶ A. 1000
- ▶ B. 500
- ▶ C. 610
- ▶ D. None of the above

Concept Test

How many ways can we arrange the letters ABCDE?

- ▶ A. 24
- ▶ B. 12
- ▶ C. 5
- ▶ D. 120

ConceptTest

What is the base case for this recursive definition?

- ▶ A. 0
- ▶ B. 1
- ▶ C. There is no base case

ConceptTest

Which of the following correctly prints a number in reverse?

▶ A.

```
void printRev1 (int n) {  
    if (n < 10)  
        printf ("%d", n);  
    else {  
        printf ("%d", n % 10);  
        printRev1 (n / 10);  
    }  
}
```

▶ B.

```
void printRev2 (int n) {  
    if (n < 10)  
        printf ("%d", n);  
    else {  
        printRev2 (n / 10);  
        printf ("%d", n % 10);  
    }  
}
```

▶ C. A and B are correct

ConceptTest

Which of the following correctly finds the length of string s?

▶ A.

```
int findLen1 (char *s) {  
    if (*s == '\0')  
        return 0;  
    else  
        return 1 + findLen1 (s+1);  
}
```

▶ B.

```
int findLen2 (char *s) {  
    if (*s == '\0')  
        return 1;  
    else  
        return 1 + findLen2 (s+1);  
}
```

▶ C.

```
int findLen3 (char *s) {  
    if (*s == '\0')  
        return 0;  
    else  
        return findLen3 (s+1) + 1;  
}
```

▶ D. Two of the above are correct

▶ E. All of the above are correct

ConceptTest

Which of the following returns true if and only if *i* is a positive even integer?

▶ A.

```
bool isEven1 (int i) {  
    if (i == 2) return true;  
    else if (i == 1) return false;  
    else  
        return isEven1 (i + 2);  
}
```

▶ B.

```
bool isEven2 (int i) {  
    if (i == 2) return true;  
    else if (i == 1) return false;  
    else  
        return isEven2 (i - 2);  
}
```

▶ C.

```
bool isEven3 (int i) {  
    if (i == 2) return true;  
    else  
        return isEven3 (i - 2);  
}
```

ConceptTest

Which of the following returns the number of occurrences of character *c* in string *s*?

- ▶ A.

```
int num1 (char *s, char c) {  
    if (*s == c)  
        return 1 + num1 (s + 1, c);  
    else  
        return 0 + num1 (s + 1, c);  
}
```

- ▶ B.

```
int num2 (char *s, char c) {  
    if (*s == '\0')  
        return 0;  
    else if (*s == c)  
        return 1 + num2 (s + 1, c);  
    else  
        return 0 + num2 (s + 1, c);  
}
```

- ▶ C. A and B are correct

ConceptTest

Consider the following code.

```
void recur (int i) {  
    if (i == 0) {  
        printf ("%d ", i);  
        return;  
    }  
    for (int j = 0; j < 2; j++)  
        recur (i - 1);  
}
```

What is printed by the call `recur (1)`?

- ▶ A. 0
- ▶ B. 0 0
- ▶ C. 0 0 0 ... infinitely
- ▶ D. 0 1
- ▶ E. None of the above

ConceptTest

Consider the following code.

```
void recur (int i) {  
    if (i <= 1) {  
        printf ("hi\n");  
        return;  
    }  
    recur (i - 1);  
    recur (i - 2);  
}
```

How many times is hi printed by the call recur (3)?

- ▶ A. 0
- ▶ B. 1
- ▶ C. 2
- ▶ D. 3
- ▶ E. 4

ConceptTest

Consider the following code.

```
void recur (int i) {  
    if (i <= 2) {  
        printf ("hi\n");  
        return;  
    }  
    recur (i - 1);  
    recur (i - 2);  
}
```

How many times is hi printed by the call recur (3)?

- ▶ A. 0
- ▶ B. 1
- ▶ C. 2
- ▶ D. 3
- ▶ E. 4

ConceptTest

We would like to count the total number of asterisks in the blob containing location (x, y) . Which of the following is the most appropriate base case for this task?

- ▶ A. If (x, y) is a blank, the total number of asterisks is 0.
- ▶ B. If (x, y) is a blank, the total number of asterisks is 1.
- ▶ C. If (x, y) is an asterisk, the total number of asterisks is 0.
- ▶ D. If (x, y) is an asterisk, the total number of asterisks is 1.

Concept Test

The following code template counts the total number of asterisks reachable from location (x, y). Select the code to use in place of the comment.

```
int blobSize (char grid[][COLS], int x, int y) {  
    int left, right, up, down;  
    if (grid[x][y] == ' ')  
        return 0;  
    else {  
        grid[x][y] = ' ';  
        left = blobSize (grid, x, y-1);  
        right = blobSize (grid, x, y + 1);  
        up = blobSize (grid, x - 1, y);  
        down = blobSize (grid, x + 1, y);  
        return ... // fill in the code  
    }  
}
```

- ▶ A. `1 + left + right + up + down;`
- ▶ B. `left + right + up + down;`
- ▶ C. `4 + right + left + up + down;`
- ▶ D. `right + left + up + down;`

ConceptTest

We've commented out some code. Is the solution still correct?

```
int blobSize (char grid[][COLS], int x, int y) {
    int left, right, up, down;
    if (grid[x][y] == ' ')
        return 0;
    else {
        //grid[x][y] = ' ';
        left = blobSize (grid, x-1, y);
        right = blobSize (grid, x+1, y);
        up = blobSize (grid, x, y-1);
        down = blobSize (grid, x, y+1);
        return 1 + left + right + up + down;
    }
}
```

- ▶ A. No, because the code could loop indefinitely
- ▶ B. No, because the grid will now be modified when the function finishes
- ▶ C. No, but I don't know why
- ▶ D. Yes

Sorting

ConceptTest

Consider the following two questions about selection sort.

- ▶ (1) Once we place a value in the sorted (green) part, can we ever move it again?
- ▶ (2) Is it ever possible to find a value in the unsorted (red) part that is smaller than **any** of the values in the sorted (green) part?

Which of the following correctly answers both questions?

- ▶ A. (1) yes, (2) yes
- ▶ B. (1) yes, (2) no
- ▶ C. (1) no, (2) yes
- ▶ D. (1) no, (2) no

ConceptTest

Consider the following two questions about insertion sort.

- ▶ (1) Once we place a value in the sorted (green) part, can we ever move it again?
- ▶ (2) Is it ever possible to find a value in the unsorted (red) part that is smaller than **any** of the values in the sorted (green) part?

Which of the following correctly answers both questions?

- ▶ A. (1) yes, (2) yes
- ▶ B. (1) yes, (2) no
- ▶ C. (1) no, (2) yes
- ▶ D. (1) no, (2) no

ConceptTest

Which of the following sorts would perform **fewer** comparisons for a large array that is already sorted? (Think of a comparison as the test of an if-statement.)

- ▶ A. Selection sort
- ▶ B. Insertion sort
- ▶ C. Both approximately equal

ConceptTest

Which of the following sorts would perform **fewer** comparisons for a large array that is sorted in reverse order? (Think of a comparison as the test of an if-statement.)

- ▶ A. Selection sort
- ▶ B. Insertion sort
- ▶ C. Both approximately equal

ConceptTest

Consider the following list of numbers:

6 2 1

What will be the list after one MI of bubble sort?

- ▶ A. 1 2 6
- ▶ B. 2 1 6
- ▶ C. 6 2 1
- ▶ D. 6 1 2

ConceptTest

Consider the following two questions about bubble sort.

- ▶ (1) Once we place a value in the sorted (green) part, can we ever move it again?
- ▶ (2) Is it ever possible to find a value in the unsorted (red) part that is larger than **any** of the values in the sorted (green) part?

Which of the following correctly answers both questions?

- ▶ A. (1) yes, (2) yes
- ▶ B. (1) yes, (2) no
- ▶ C. (1) no, (2) yes
- ▶ D. (1) no, (2) no

Structs and Malloc

ConceptTest

```
int partition (int low, int high, int pivot, int a[]) {  
    int i = low, j = low;  
    while (j <= high) {  
        if (a[j] < pivot) {  
            swap (a, i, j);  
            i++;  
        }  
        j++;  
    }  
    return i;  
}
```

Assume we partition the array [6, 8, 3] around pivot 4. What is the resulting array?

- ▶ A. [3, 6, 8]
- ▶ B. [8, 6, 3]
- ▶ C. [3, 8, 6]

ConceptTest

```
struct name {  
    char firstName[20];  
    char lastName[20];  
};
```

```
struct student {  
    struct name studentName;  
    int year;  
};
```

```
struct student s;
```

How do I change the first letter of the first name of s?

- ▶ A. `s.name.firstName[0] = 'D';`
- ▶ B. `s.studentName.firstName[0] = 'D';`
- ▶ C. `s.studentName[0] = 'D';`
- ▶ D. `s.firstName[0] = 'D';`

Concept Test

```
struct student {  
    char firstName[20];  
    char lastName[20];  
    int year;  
};
```

Consider the following function prototype:

```
void proc (struct student s1, struct student s2, char *p);
```

Also assume that a and b have been declared as `struct student` and have been initialized.

Without worrying about what the function will do, which function call is valid?

- ▶ A. `proc (a, b, "go");`
- ▶ B. `proc (a, a, "go");`
- ▶ C. `proc (a, a, b.firstName);`
- ▶ D. A and B
- ▶ E. All of the above

ConceptTest

Assume we want to store information about 100 people. For each person, we will store their first name, last name, and age. How should we store this data?

- ▶ A. As an array, each of whose elements is a structure
- ▶ B. As a structure, each of whose components is an array
- ▶ C. As an array, each of whose components is a string
- ▶ D. As a structure, each of whose components is a string

ConceptTest

We want to store one student's first name, last name, age, and marks they earned in 100 courses. What should we use?

- ▶ A. An array, each of whose elements is a structure with four components
- ▶ B. A structure with four components, one of which is an array of ints
- ▶ C. An array, each of whose components is a string
- ▶ D. A structure, each of whose components is a string

ConceptTest

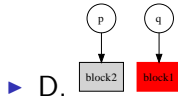
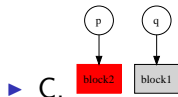
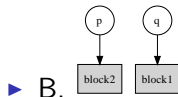
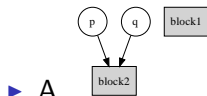
We would like to write a function that takes two struct parameters `from` and `to`, and copies the values in `from` to the values in `to`. What types of parameters should the function take?

- ▶ A. Two structs
- ▶ B. Two pointers to structs
- ▶ C. One pointer to struct, and one struct

ConceptTest

What is the result of the following code? (A red box indicates a block of memory that has been freed.)

```
char *p = malloc(5);  
char *q = p;  
p = malloc(5);
```



ConceptTest

```
struct student {  
    char firstName[20];  
    char lastName[20];  
    int year;  
};
```

What is the best way to dynamically allocate a struct student?

- ▶ A. `struct student *s = malloc(struct student);`
- ▶ B. `struct student *s = malloc(sizeof(struct student));`
- ▶ C. `struct student *s = malloc(44);`