

Started on Monday, October 31, 2022, 1:56 PM

State Finished

Completed on Monday, October 31, 2022, 1:57 PM

Time taken 39 secs

Question **1**

Correct

Points out of
1.00

The address of operator `&` returns both the address and the value of its operand.

Select one:

- ☐ True
- ☒ False ✓

Question **2**

Correct

Points out of
2.00

If `i` is a variable and pointer variable `p` points to `i`, which of the following expressions are aliases for `i`?

Select one or more:

- ☐ `*&p`
- ☒ `*p` ✓
- ☐ `&*i`
- ☐ `*i`
- ☐ `&*p`
- ☒ `*&i` ✓
- ☐ `&i`
- ☐ `&p`

Question **3**

Correct

Points out of
2.00

If `i` is an `int` variable and `p` and `q` are pointers to `int`, which of the following expressions are legal?

Select one or more:

- ☐ `&p = q`
- ☐ `p = &q`
- ☐ `p = i`
- ☐ `p = *q`
- ☒ `p = q` ✓
- ☒ `p = *&q` ✓
- ☐ `*p = q`
- ☒ `*p = *q` ✓
- ☐ `*p = &i`

Question 4

Correct

Points out of
2.00

Given the following definition statement

```
int x = 10, *p = &x, *q = p;
```

which of the following expressions are valid?

Select one or more:

- ☐ `p = x`
- ☒ `*p = 56` ✓
- ☒ `*p = *q` ✓
- ☒ `q = &x` ✓
- ☒ `p = q` ✓
- ☐ `*p = q`

Question 5

Correct

Points out of
1.00

Given the first definition statement

```
int x = 10;
```

is the following second subsequent definition statement valid?

```
int* y = &x, z = &x;
```

Select one:

- ☐ True
- ☒ False ✓

Question 6

Correct

Points out of
1.00If `p` is a pointer variable, then the statement

```
p = p * 2;
```

is valid.

Select one:

- ☐ True
- ☒ False ✓

Question 7

Correct

Points out of
2.00

Given the following code fragment

```
int x;
int *y = &x;
x = 10;
```

what is the value resulting from the following expression's evaluation? If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value obtained after the evaluation of the expression.

```
x**y*x+y
```

Answer: 1010 ✓

Question 8

Correct

Points out of
1.00

Given the following definition statement

```
int x, *y = &x;
```

the following expression

```
x**y*x+y
```

contains 9 ✓ tokens.

Question **9**

Correct

Points out of
1.00

Given the definitions

```
int array[1000];  
int *pa;
```

is the following expression valid?

```
pa = array
```

Select one:

☒ True ✓

☐ False

Question **10**

Correct

Points out of
1.00

Given the following definition statement:

```
char a[] = "Digipen";
```

consider the expression:

```
sizeof(a)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value obtained after the evaluation of the expression.

Answer: 8 ✓

Question **11**

Correct

Points out of
1.00

Given the following definition:

```
char b[] = {'D', 'i', 'g', 'i', 'p', 'e', 'n'};
```

consider the expression:

```
sizeof(b)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value obtained after the evaluation of the expression.

Answer: 7 ✓

Question **12**

Correct

Points out of
1.00

Given the following definition:

```
char c[] = {'D', 'i', 'g', '\0', 'i', 'p'};
```

consider the expression:

```
sizeof(c)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value obtained after the evaluation of the expression.

Answer: 6 ✓

Question **13**

Correct

Points out of
1.00

Given the following definition:

```
char *d = "Digipen";
```

consider the expression:

```
sizeof(d)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value obtained after the evaluation of the expression.

Answer: 8

Question **14**

Correct

Points out of
1.00

Given the following definition:

```
char a[] = "Digipen";
```

consider the expression

```
/*  
strlen is the standard library function declared in <string.h> that  
was discussed in lectures.  
*/  
strlen(a)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value obtained after the evaluation of the expression.

Answer: 7

Question **15**

Correct

Points out of
1.00

Given the following definition:

```
char b[] = {'D', 'i', 'g', 'i', 'p', 'e', 'n'};
```

consider the expression:

```
/* strlen is a standard library function declared in <string.h> */  
strlen(b)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value obtained after the evaluation of the expression.

Answer: UDB

Question **16**

Correct

Points out of
1.00

Given the following definition:

```
char c[] = {'D', 'i', 'g', '\0', 'i', 'p'};
```

consider the expression:

```
/* strlen is a standard library function declared in <string.h> */  
strlen(c)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value obtained after the evaluation of the expression.

Answer: 3



Question 17

Correct

Points out of
1.00

Given the following definition:

```
char *d = "Digipen";
```

consider the expression:

```
/* strlen is a standard library function declared in <string.h> */
strlen(d)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value obtained after the evaluation of this expression.

Answer: 7



Question 18

Correct

Points out of
2.00

Consider the following code fragment involving an array:

```
char str[] = "UnCopyRightAbles";
char *p = str + 5, ch = (*p)++;
printf("%c,%s", ch, str);
```

If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the exact values printed to standard output by the **printf** statement.

Answer: y,UnCopzRightAbles



Question 19

Correct

Points out of
2.00

Consider the following code fragment involving an array:

```
char str[] = "Mathematical";
char *p = str + 7, ch = (*p)--;
printf("%c,%s", ch, str);
```

If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the exact values printed to standard output by the **printf** statement.

Answer: t,Mathemasical



Question 20

Correct

Points out of
2.00

Consider the following code fragment involving an array:

```
char str[] = "bresenhams";
char *p = str + sizeof(str) - 8, ch = *++p;
printf("%c,%s", ch, str);
```

If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the exact values printed to standard output by the **printf** statement.

Answer: e,bresenhams



Question 21

Correct

Points out of
2.00

Consider the following code fragment involving an array:

```
char str[] = "Multidimensional";
char *p = str + sizeof(str) - 1, ch = *--p;
printf("%c,%s", ch, str);
```

If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the exact values printed to standard output by the **printf** statement.

Answer: l,Multidimensional



Question 22

Correct

Points out of
2.00

Consider the following code fragment involving an array:

```
char str[] = "CppTemplates";
char *p = str + 5, ch = *p++;
printf("%c,%s", ch, p);
```

If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the exact values printed to standard output by the **printf** statement.

Answer: 

Question 23

Correct

Points out of
2.00

Consider the following code fragment involving an array:

```
char str[] = "GameDesigner";
char *p = str + sizeof(str) - 6, ch = *p--;
printf("%c,%s", ch, p);
```

If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the exact values printed to standard output by the **printf** statement.

Answer: 

Question 24

Correct

Points out of
2.00

Consider the following code fragment involving an array:

```
char str[] = "DataStructures";
char *p = str + sizeof(str) - 2, ch = ++*p;
printf("%c,%s", ch, str);
```

If code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the exact values printed to standard output by the **printf** statement.

Answer: 

Question 25

Correct

Points out of
2.00

Consider the following code fragment involving a character array:

```
char str[] = "ComputerGraphics";
char *p = str + 5, ch = --*p;
printf("%c,%s", ch, str);
```

If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the exact values printed to standard output by the **printf** statement.

Answer: 

Question 26

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 };
int *p = a + 2;
```

Now, consider the expression:

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 

Question **27**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 2;
```

Now, consider the expression:

```
-3[p]
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: -9

Question **28**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 5;
```

Now, consider the expression:

```
p[-3]
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 3

Question **29**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 };
int *p = a + sizeof(a)/sizeof(a[0]);
```

Now, consider the expression:

```
-p[-3]
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: -4

Question **30**

Correct

Points out of
1.00

Consider the following declaration statement:

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 2;
```

Now, consider the expression:

```
*(a*p)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 2



Question 31

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 };
int *p = a + 2;
```

Now, consider the expression:

`&p[4]`

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 124



Question 32

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 };
int *p = a + 7;
```

Now, consider the expression:

`(p - 6)`

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 104



Question 33

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 };
int *p = a + 5;
```

Now, consider the expression:

`*p++`

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 9



Question 34

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 6;
```

Now, consider the expression:

`(*p)++`

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 0



Question **35**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 };
int *p = a + a[7];
```

Now, consider the expression:

```
*++p
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer:

Question **36**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 };
int *p = a + 2;
```

Now, consider the expression:

```
++*p++
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer:

Question **37**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 };
int *p = a + 6;
```

Now, consider the expression:

```
(*++p)++
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer:

Question **38**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + a[6];
```

Now, consider the expression:

```
*(p+(p+4))
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer:



Question **39**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 4;
```

Now, consider the expression:

```
p+a[5]
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 152

Question **40**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 8;
```

Now, consider the expression:

```
p[-a[3]]
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 0

Question **41**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 2;
```

Now, consider the expression:

```
*(p+a[3])
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 1

Question **42**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 5;
```

Now, consider the expression:

```
*p+a[3]
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 11



Question **43**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 2;
```

Now, consider the expression:

p-5

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 88

Question **44**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 2;
```

Now, consider the expression:

p + 3

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 120

Question **45**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 2;
```

Now, consider the expression:

***p+3**

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 6

Question **46**

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + a[2];
```

Now, consider the expression:

5[p]

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 7



Question 47

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 2;
```

Now, consider the expression:

```
&p
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 200



Question 48

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a + 2;
```

Now, consider the expression:

```
p[-2]
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 5



Question 49

Correct

Points out of
1.00

Assume the 64-bit compiler assigns storage for array object **a** and variable **p** at memory addresses 100 and 200, respectively.

```
int a[] = { 5, 8, 3, 2, 1, 9, 0, 4, 7, 6 }, *p = a - 2;
```

Now, consider the expression:

```
*p + 5
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: UDB



Question 50

Correct

Points out of
1.00

In the following definitions, assume the 64-bit compiler assigns storage for array object **array** at address 1000.

```
short array[] = { 3, 6, 2, 4, 7, 8 }, *p1 = array + 1, *p5 = array + 5;
```

Now, consider the expression:

```
p5 - p1
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 4



Question **51**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };  
short *p1 = array + 1, *p5 = array + 5;
```

Now, consider the expression:

p5 + p1

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: CTE

Question **52**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };  
short *p1 = array + 1, *p5 = array + 5;
```

Now, consider the expression:

p5 - --p1

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 5

Question **53**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };  
short *p1 = array + 1, *p5 = array + 5;
```

Now, consider the expression:

p5 - p1--

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 4

Question **54**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };  
short *p3 = array + 3;
```

Now, consider the expression:

p3+2 = &array[2]

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: CTE



Question **55**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };  
short *p3 = array + 3;
```

Now, consider the expression:

```
*(p3+2) = 6
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: Question **56**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };  
short *p3 = array + 3;
```

Now, consider the expression:

```
*(p3++) = 5
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: Question **57**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };  
short *p3 = array + 3;
```

Now, consider the expression:

```
*--p3 = *(array+4)
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: Question **58**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };  
short *p1 = array + 1, *p5 = array + 5;
```

Now, consider the expression:

```
p1 - p5
```

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 

Question **59**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };
short *p1 = array + 1, *p3 = array + 3, *p5 = array + 5;
```

Now, consider the expression:

p1 - p3 + p5

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 1006

Question **60**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };
short *p1 = array + 1, *p3 = array + 3, *p5 = array + 5;
```

consider the expression:

p1 - (p3 - p5)

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 1006

Question **61**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };
short *p1 = array + 1, *p3 = array + 3, *p5 = array + 5;
```

consider the expression:

p1 - p3 - p5

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: CTE

Question **62**

Correct

Points out of
1.00

Given the following definitions, assume the **64**-bit compiler assigns storage for array object **array** at address **1000**.

```
short array[] = { 3, 6, 2, 4, 7, 8 };
short *p1 = array + 1, *p5 = array + 5;
```

Now, consider the expression:

p5 - 2 - p1

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 2



Question 63

Correct

Points out of
1.00

Given the following definitions, assume the 64-bit compiler assigns storage for array object **array** at address 1000.

```
short array[] = { 3, 6, 2, 4, 7, 8 };
short *p1 = array + 1, *p3 = array + 3, *p5 = array + 5;
```

Now, consider the expression:

p1 += p5 - p3

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 1006



Question 64

Correct

Points out of
1.00

Given the following definitions, assume the 64-bit compiler assigns storage for array object **array** at address 1000.

```
short array[] = { 3, 6, 2, 4, 7, 8 };
short *p1 = array + 1, *p3 = array + 3, *p5 = array + 5;
```

Now, consider the expression:

p1 -= p5 - p3

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 998



Question 65

Correct

Points out of
1.00

Given the following definitions, assume the 64-bit compiler assigns storage for array object **array** at address 1000.

```
short array[] = { 3, 6, 2, 4, 7, 8 };
short *p1 = array + 1;
short *p5 = array + 5;
```

Now, consider the expression:

p1 = p5 - 3

If the expression cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value resulting from the expression's evaluation.

Answer: 1004



Question 66

Correct

Points out of
4.00

Consider the definition of function **foo**:

```
int mystery(char const *src) {
    char const *pc = src;
    while (*src) src++;
    return src-pc;
}
```

What does the following function call expression evaluate to?

mystery("subdermatoglyphic")

If the expression is illegal or cannot be compiled, write CTE (for compile-time error). If the expression generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the value that the expression evaluates to.

Answer: 17



Question 67

Correct

Points out of
4.00

If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the values printed to standard output.

```
void mystery(int *a, int s) {
    int *b = a + s;
    while (++a != b) {
        *a += *(a - 1);
    }
}

int g[] = { 1, 2, 3, 4, 5, 6 }, *b = g;
int *p = g + sizeof(g)/sizeof(g[0]);
mystery(g, 3);
mystery(g + 2, 3);
while (b != p) {
    printf("%d,", *b++);
}
```

Answer: 1,3,6,10,15,6,



Question 68

Correct

Points out of
4.00

Write the comma-separated 16-bit hexadecimal addresses printed to standard output by the following code fragment. Assume the 64-bit compiler provides storage to object `bart` at address `0x0100`.

```
int bart, *p = &bart;
for (bart = 0; bart < 4; ++bart) {
    printf("%p%s", (void*)(p+bart), (bart==3)?"":",");
}
```

Answer: 0x0100,0x0104,0x0108,0x010C



Question 69

Correct

Points out of
4.00

Write the sequence of characters printed to standard output by the following code fragment. If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the output printed to standard output.

```
char str[] = "CapeOfGoodHope", *p = 5+str;
while (p >= str) {
    ++*p;
    --p;
}
fputs(str, stdout);
```

Answer: DbqfPgGoodHope



Question 70

Correct

Points out of
4.00

Write the sequence of characters printed to standard output by the following code fragment. If the code fragment cannot be compiled, write CTE (for compile-time error). If the code fragment generates undefined behavior (see pages 65 and 163 of text), write UDB (for undefined behavior). Otherwise, write the output printed to standard output.

```
char str[] = "DigiPen", *p;
for (p = str+strlen(str)-1; p >= str; --p) {
    ++*p;
}
fputs(str, stdout);
```

Answer: EjhjQfo



Question 71

Correct

Points out of
1.00

Adding 1 to a pointer increases the address stored in it by 1 byte.

Select one:

- ☐ True
- ☒ False ✓

Question **72**

Correct

Points out of
1.00

It is a compiler error to use pointer arithmetic with a pointer that does not reference an array.

Select one:

- ☐ True
- ☒ False ✓

Question **73**

Correct

Points out of
1.00

It is a compiler error to subtract two pointers that are referencing different arrays.

Select one:

- ☐ True
- ☒ False ✓

[◀ Lecture 16: Pointer Arithmetic; Pointers
and Arrays](#)

Jump to...

[Lab 8: Arrays and Characters ▶](#)