CS-1714-0B1-Fall-2020-Computer Programming II
ABOUT
Course Info
Teaching Staff
Syllabus & Schedule
COURSEWORK
Content Modules
Programming Setup
Projects
Exams
Grades
DIRECT LINKS FOR ALL MODULES
PlayPosit Bulbs - Recorded Videos
Lecture Slides
Quizzes
COMMUNICATION
Live Sessions - Blackboard Collaborate Ultra
Tutoring
Announcements
HELP
Student Support
Blackboard Help
Tools

Review Test Submission: Quiz-03

User	Rod Miles Simms
Course	CS-1714-0B1-Fall-2020-Computer Programming II
Test	Quiz-03
Started	9/13/20 5:39 PM
Submitted	9/13/20 6:14 PM
Due Date	9/13/20 11:59 PM
Status	Completed
Attempt Score	7 out of 10 points
Time Elapsed	35 minutes out of 1 hour
Results Displayed	All Answers, Submitted Answers, Correct Answers, Feedback, Incorrectly Answered Questions

Question 1

1 out of 1 points

Which of the following `printf` statements print out the memory address that the pointer variable `ptr` is point at?

Selected Answer: ✓ `printf( "%p", ptr );`

Answers:
- `printf( "%d", *ptr );`
- ✓ `printf( "%p", ptr );`
- `printf( "%d", (void *) ptr );`
- `printf( "%lf", ptr );`

Question 2

0 out of 1 points

What are the contents of the numbers array at the end of this code?

```
int numbers[] = { 35, 57, 78, 66, 41, 12 };
int *ptrA = numbers;
int *ptrB = ptrA + 2;
*ptrB = 100;
*ptrA = 100;
```

Selected Answer: ✗ `35, 57, 78, 66, 41, 12`

Answers:
- `100, 100, 78, 66, 41, 12`
- ✓ `100, 57, 100, 66, 41, 12`
- `35, 57, 78, 66, 41, 12`
- `0, 0, 0, 0, 0, 0`

Question 3

1 out of 1 points

What is a pointer in C?

Selected Answer: ✓ A variable that holds the memory address of another variable

Answers:
- ✓ A variable that holds the memory address of another variable
- A variable that holds multiple values
- A special variable that can be used to represent Boolean values in C
- A primitive data type

Question 4

1 out of 1 points

From where is memory dynamically allocated?

Selected Answer: ✓ heap

Answers:
- Harddrive
- ✓ heap
- Static
- stack

Question 5

0 out of 1 points

What is the value of `*ptr` after this code block?

```
int numbers[] = { 35, 57, 78, 66, 41, 12 };
int *ptr = numbers + 5;
```

Selected Answer: ✗ 41

Answers:
- 41
- The memory address of the 5th item in the `numbers` array
- The length of the array
- ✓ 12

Question 6

0 out of 1 points

What are the contents of `array` after this code?

```
 6    int array[ 6 ] = { 10, 20, 30, 40, 50, 60 };
 7    int i;
 8    for( i = 0; i < 6; i++ )
 9    {
10        if( i % 2 == 1 )
11            *( array + i ) = 3;
12    }
13    *array = 100;
```

Selected Answer: ✗ 10, 20, 30, 40, 50, 60

Answers:
- 10, 3, 30, 3, 50, 100
- ✓ 100, 3, 30, 3, 50, 3
- 100, 20, 3, 40, 3, 60
- 10, 3, 30, 3, 50, 3
- 10, 20, 30, 40, 50, 60

Question 7

1 out of 1 points

What does it mean if `malloc()` returns a `NULL`?

Selected Answer: ✓ It was unable to dynamically allocate memory

Answers:
- If `malloc()` returns NULL, you must use `calloc()`
- NULL is the first available memory address
- ✓ It was unable to dynamically allocate memory
- It was successful in allocating memory dynamically

Question 8

1 out of 1 points

What is the output of the following code?

```
int numbers[] = { 35, 57, 78, 66, 41, 12 };
int *ptrA = numbers;
int *ptrB = ptrA + 2;
int *ptrC = ptrB + 1;
printf( "%d, %d, %d\n", *ptrA, *ptrB, *ptrC );
```

Selected Answer: ✓ `35, 78, 66`

Answers:
- `66, 78, 41`
- `35, 78, 57`
- ✓ `35, 78, 66`
- `66, 35, 41`

Question 9

1 out of 1 points

Which of the following statements correctly dynamically allocates an array for 10 integers?

Selected Answer: ✓ `int *m = (int *) malloc( 10 * sizeof( int ) );`

Answers:
- ✓ `int *m = (int *) malloc( 10 * sizeof( int ) );`
- `int *m = (int *) malloc( 10 * int );`
- `int *m = (int *) malloc( 10 );`
- `int *m = (int *) malloc( int ) * 10;`

Question 10

1 out of 1 points

Which of the following `printf` statements print out the value inside the memory location that the pointer variable `ptr` is point at?

Selected Answer: ✓ `printf( "%d", *ptr );`

Answers:
- `printf( "%lf", ptr );`
- ✓ `printf( "%d", *ptr );`
- `printf( "%d", (void *) ptr );`
- `printf( "%p", (void *) ptr );`

Wednesday, September 30, 2020 9:49:52 AM CDT

← OK