**CSCD 240 Fall 2020 Name: Ian Kaiserman**

**Quiz 3 Date: Nov 4 2020**

**---------------------------------------------------------------------------------------------------------------------**

**Short Questions:**

**--------------------------------------------------------------------------------------------------------------**

1. **What will the following program print?**

#include <stdio.h>

int main(){

int v = 10;

int \*p = &v;

\*p = 40;

printf("v is %d\n", v);

return 0;

}

**40**

--------------------------------------------------------------------------------------------------------------

1. **In the following program, what will be the value of ‘b’?**

int c = 10;

int \*ptr = &c;

int a = \*ptr;

\*ptr = 50;

int b = \*&a;

**10**

--------------------------------------------------------------------------------------------------------------

1. **In the following program, what will be the value of ptr[2]?**

int array[10];

int i;

for (i = 0; i < 10; i++)

array[i] = i;

int \*ptr;

ptr = &array[5];

**7**

---------------------------------------------------------------------------------------------------------------------

1. **Consider the following array:**

int a[4] = {8, 9, 5, 6};

int \*ptr = a;

Assume the base address of array **a** is **0x1000**. What will be printed by **(ptr+2)**?

**0x1008**

-----------------------------------------------------------------------------------------------------------------

1. In the following program, **modifyElement** function cannot modify the value of a[3]. Modify the **modifyElement** function so that the value of **a[3]** can be changed. What will be the new value of **a[3]**?

void modifyElement(int p);

int main(){

int a[] = { 0, 1, 2, 3, 4 };

modifyElement( a[3]);

:

:

}

void modifyElement(int p){

p \*=10;

}

**void modifyElement(int p[], int i);**

**int main(){**

**int a[] = { 0, 1, 2, 3, 4 };**

**modifyElement( a, 3);**

**:**

**:**

**}**

**void modifyElement(int p[], int i){**

**p[i] \*=10;**

**}**

**----------------------------------------------------------------------------------------------------------------**

**6.** What is the difference between **++\*ptr** and **\*ptr++** ? Explain in the context of the following expressions.

int i;

int array[SIZE];

for (i = 0; i < SIZE; i++)

array[i] = i;

int \*ptr;

ptr = &array[0];

**\*ptr++ increments ptr to the next element of array, while ++\*ptr increases the value of the element ptr is pointing to currently and doesn’t increment.**

**In the above code, \*ptr++ increments ptr from array[0] to array[1], and ++\*ptr increases array[0] by 1 and keeps the pointer position at array[0].**

**7.** Explain **‘ptr’**, **‘\*ptr’**, **‘&ptr’** in the context of the following code snippet.

int i = 10;

int \*ptr = &i;

**ptr is the raw address of the value being pointed at, currently pointing to the address of i. \*ptr is the dereferenced value of what ptr is pointing to, in this case the value of i. &ptr is the raw address of the POINTER, rather than the variable being pointed at, so it is the address of ptr.**

---------------------------------------------------------------------------------------------------------------------------

8. Is there any problem in the following program? How will you fix it?

int array[SIZE];

int i;

for (i = 0; i < SIZE; i++)

array[i] = i;

int \*ptr= array;

for (i = 0; i < SIZE; i++)

printf("%d ", \*array++);

**Incrementing arrays with ++ can’t be done with raw array references. The pointer to array should be used as that syntax is correct in that case. “\*ptr++”**

--------------------------------------------------------------------------------------------------------------------------

**9.** A C program contains the following statements.

char v = ‘A’;

char \*pv = &v;

\*pv = v +1;

What value is assigned to ‘v’? What value is represented by \*pv?

**v is assigned the ASCII value of ‘A’. the value represented by \*pv is the ASCII value of ‘B’, as that is the ASCII value of ‘A’ + 1.**

-----------------------------------------------------------------------------------------------------------------------

10. Consider the following program. What values will be displayed by the **‘printf’** statements?

int a = 10;

int b = 20;

int \*ptr = &a;

ptr = &b;

\*ptr = 30;

int total = \*ptr + 3;

printf(“%d”, b);

printf(“%p”, ptr);

printf(“%d”, \*ptr);

printf(“%d”, total);

**30**

**Address of ptr**

**30**

**33**

**Identify whether the following statements are true or false.**

1. Data held by a pointer is the address of a location in memory.
   1. **True**
2. The line marked with an arrow is true.

char x[16];

char \*p = x;

x[1] is equivalent to (p+1). 🡨

**True**

1. In the following code snippet, the line marked with an arrow is a valid expression.

int \*ptr = &i;

scanf(“%d”, &ptr); 🡨

**False**

1. Consider the following piece of code. The line with an arrow means that three times its scalar value will be added to ‘ptr’.

int x[5];

int \*ptr = x;

ptr =ptr + 3; 🡨

**False**

1. A void pointer can be freely assigned to and compared with any type of pointer without an explicit cast being required.
   1. **True**
2. As an array is passed to a function, it loses its size.
   1. **True**
3. Any function written to receive an array argument is given the address of the first element of the array passed to it.
   1. **True**
4. An array cannot be assigned but a pointer can be.
   1. **False**
5. Memory allocation function **realloc()** can only grow the memory block granted previously by a successful call to **malloc()**, **calloc()**, or **realloc()**.
   1. **True**
6. Considering the following expressions where ‘**ptr**’ is a constant pointer, the expression in the line with an arrow is correct.

char c[10];

char \*const ptr = &c[0];

int k = \*(ptr+1); 🡨

**False**