Q1

a. 5␣␣,␣6.266
b. |1000,␣15|
c. NC: *format specifies 'int' but the argument is 'char *'*
d. abcd,␣

Q2

a. 'printf' returns the number of characters printed. If there are errors, 'printf' will return a negative value.

Q3

a. x = 2, y = AMBI, z = 1
b. a = 6, b = AMBI
c. str = "hello", c = 32 (or '␣'), x = 6, y = 7.0f

Q4

a. NC: *switch statement requires expression of integer type*
b. NC: *switch case must be an integer constant expression*
c. hello

Q5

a.

```
i = 0;
while (i < 7) {
    j = i + 1;
    while (j < 7) {
        if ((j + i) % 3 == 0) {
            printf("%d-%d ",i,j);
        }
        ++j;
    }
    ++i;
}
```

b. 0-3␣0-6␣1-2␣1-5␣2-4␣3-6␣4-5␣ (assuming i and j are defined)

Q6

a. Is␣it␣me?:␣0

Q7

a. 8,␣5,␣1

Q8

a. 2␣6␣3␣8␣15␣

Q9

a. 1:␣5,␣1,␣3,␣0,␣6
b. 1:␣1,␣1,␣2,␣1,␣3
c. 1:␣2,␣1,␣3,␣0,␣4

(note: can't compile if -Werror is used.
'&&' within '||' and *expression result unused*)

Q10

a. (x > 6) ? (y += 7) : ((x < 3) ? (y -= 8) : 1);
   (note: 1 can be substituted with anything)
   y += (x > 6) ? 7 : ((x < 3) ? -8 : 0);
   (alternative answer)

Q11

a. (i < 3) && (i += 2);
b. (x <= y) && (x = y + 1);
c. (x < y) && (y = 5) || (y = x);

Q12

a. int Foo (int a, float b);
b. void Boo (int *a);
c. const float* Goo (void);
   (alternatively)
   float const* Goo (void);
d. void Doo (int *const a);

e. void Hoo (double *const *a);
f. void Joo (float (*a)[6]);
g. void Loo (float (*const a)[]);

Q13 (1)

a. int Foo[6] = { 0 };
b. int Goo[8] = { 1, 3, 5, 6 };
c. int Boo[100];
   int i = 0;
   for (i = 0; i < 100; ++i)
       Boo[i] = 2;
d. int Goo[2][3] = {{ 1, 2, 0 }, { 3, 0, 0 }};
e. int Loo[100][10];
   int i = 0;
   for (i = 0; i < 1000; ++i)
       Loo[0][i] = i;

Q13 (2)

'ptr' is a pointer to an array of integers.
The size of int[] is unknown so you cannot perform (int[])* arithmetic.
(i.e. the size of 'int[5]' = sizeof(int) * 5 but size of 'int[]' = ???)
NC: *Arithmetic on a pointer to an incomplete type 'int []'.*

Q14

a. int *
b. int **
c. int
d. int *
e. 1. NC
   2. int[3]
f. int *
g. int
h. int(*)[3]
i. int[2][3]
j. int *
k. int(*)[3]
l. int(*)[2][3]

Q15

a. -3
b. NC
c. 0
d. NC
e. 1006
f. NC
g. 1004
h. 1004
i. 998
j. 770 (or 0x0302)
k. 2
l. -9
m. NC

Q16

(note: int i is not an intended error)

a. cs120isdead

Q17

a. NC: *cannot increment value of type 'int [6]' (a.k.a. array)*

Q18

a. Hello, Hellofalicious

Q19

(note: assuming stdlib is included)

a. RTE: *Double Free and Dangling Pointer. Pointer is still pointing at an address that has been freed and tried to free that address again*
b. 8 bytes

Q20

a. C
b. C
c. NC: *cannot assign to variable 'ipc' with const-qualified type 'int *const'*

d. NC: cannot assign to variable 'ipc' with const-qualified type 'int *const'
e. NC: *variable 'ip' is uninitialized when used here*
f. NC: *assigning to 'int *' from 'const int *' discards qualifiers*
g. C
h. C (NC with -Werror: *expression result unused*)
i. NC: *read-only variable is not assignable*
j. C

Q21

You cannot declare a struct of the same type within the struct

Q22

(need TA to check)

Q23

a. 6
b. 32
c. 8
d. (need TA to check)
e. (need TA to check)
f. (need TA to check)
g. (need TA to check)
h. (need TA to check)
i. (need TA to check)

Q24

a) 1. Should not return the address of the local variable. 2. Cannot declare a variable size array

b)

December 2, 2019

xWillFlame Today at 12:05 AM

```
int* CreateArray(int num)
{
    int a* =  (int*)malloc(sizeof(int) * num);
    return a;
}
```

Q25

a)

```
int** Create2DArray(int width, int height)
{
    int** arr = (int**)malloc(sizeof(int*) * height);
    int i = 0;
    for(int i = 0; i < height; ++i)
    {
        arr[i] = (int*)malloc(sizeof(int) * width);
    }
    return arr;
}
```

b)

```
void FreeArray(int** arr, int height)
{
    for(int i = 0; i < height; ++i)
        free(arr[i]);
    free(arr);
}
```

Q26

```
int strlen (char* str)
{
    int i = -1;
    while(str[++i] != 0);
    return i;
}
```

Q27

```
char* ConcatenateNewString(char* a, char* b)
{
    char* result = (char*)malloc(strlen(a) + strlen(b) + 1);
    char* tmp = result;

    do
    {
        *tmp++ = *a++;
    }while(*a != 0);

    do
    {
        *tmp++ = *b++;
    }while(*b != 0);

    *tmp = 0;
    return result;
}
```

Q28

xWillFlame Today at 12:55 AM

```
void Swap(int *a, int size)
{
    int* tmpP = a + size - 1;
    int tmp = 0;

    while(a < tmpP)
    {
        tmp = *a;
        *a++ = *tmpP;
        *tmpP-- = tmp;
    }
}
```

```
void Swap(int *a, int size)
{
    int* tmpP = a + size - 1;

    while(a < tmpP)
    {
        *a = *a + *tmpP;
        *tmpP = *a - *tmpP;
        *a = *a - *tmpP;
        a++;
        tmpP--;
    }
}
```