10/26/2025

CS 211

QUIZ 3

(1) What is the output of C Program?

```
#include <stdio.h>

int main() {

    int a[];

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

    printf("%d\n", a[0]);

}
```

illegal → compiler error

(2) Output of following program?

```
#include <stdio.h>

void fun (int** ptr) {
    **ptr = 30;
}

int main() {

  int y = 20;
  int* pointer = &y;
  fun ( &pointer ) ;

  printf("%d\n", y);

  return 0;

}
```

updated to 30; ptr; ptr to ptr; prints new value: 30

(3) You declare and define the following function:

```
int quizSwap ( int a, int* b ) {
    int temp = *b;
    *b = a;
    a = temp;
    return a;
}
```

ptr; data of b, 12; b = 4; a = 12; return 12

Then, you run the following snippet of code. What will be printed to the command line?

```
int a = 4;
int b = 12;
int c = quizSwap ( a, &b );
printf ( "%d\n", a+b+c );
```

ptr; after func: a = 4, b = 4, c = 12; 4+4+12 = 20

prints 20

(4) Here is a short program:

```
#include <stdlib.h>

int main () {
    int numberVar;
    int* pointerVar = malloc(sizeof(int));
    free(pointerVar);
}
```

if you need to dynamically allocate it: it's in heap, otherwise it's in Stack.

empty var, 4 bytes; empty ptr, 4 bytes; freed ptr

According the C memory model, where are numberVar and *pointerVar stored in the memory?

numberVar in Stack, pointerVar in heap

(5) Here is a short program:

```
#include <stdlib.h>

typedef struct quiz {
  float* data;
} quiz_t;

int main () {
  quiz_t* myQuizType = malloc(sizeof(quiz_t));
  ???? thingVar = myQuizType->data;
}
```

if this was quiz_t then myQuizType.data would be correct and myQuizType->data would need a float not a float*

ptr; *(myQuizType).data

What is the correct data type in the ???? for thingVar?

float*

(6)

```
#include <stdlib.h>
#include <stdio.h>

typedef struct node {
    char job;
    struct node* next;
} Node;

int main() {

    Node myNode = { .job='A', .next=NULL };

    printf( "%c\n", myNode.next->job );

}
```

(*myNode).job; dereferences next which is null.

Error, dereferencing NULL pointer

(7) What is the result of this program:

```
#include <stdio.h>

int main() {

  int x[1] = {10};

  printf("%d\n", x[-1]);

  return 0;

}
```

illegal, stack buffer underflow