10/25/2025

CS 211 Quiz 1

(3)

You have compiled the following C program into the program "printArgs":

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

int main(int argc, char* argv[]) {
  printf ( "%s\n", argv[2] );
  return EXIT_SUCCESS;
}
```

↳ 0,1,2 so takes the 3rd argument

Then, you run the following command:

```
./printArgs 2 3 4
```

↳ arg1 1 2 2 3

What would the program print to the command line as output?

It will output the 3rd argument which is 3.

(4)

The control flow statements in C, including "do," "while," "continue," and "break" function very similarly to that in Java, and are widely adopted in many programming languages influenced by C. You have compiled this program. What will it print out?

```
#include <stdio.h>

int main () {
  int number = 0;

  do {
    if (number==0) {
      number=1;
      continue;
    }
    if (number%4==0) {
      break;
    }
    number++;
  } while (number<5);

  printf("%d\n", number);
}
```

1st iter 2nd iter 3rd iter 4th iter 5th iter

int number = 0; → 0 → → 2 → 3 → 4

do { → 1 → 2 → 3 → 4

if (number==0) { → true → false → false → false → false

number=1; → 1

continue; → ends iter

if (number%4==0) { → false → false → false → true

break; → ends do while loop

number++; → 2 → 3 → 4

} while (number<5); → true → true → true

printf("%d\n", number); → 4

(5)

The meaning of the static keyword in Java was inspired by its predecessor language, C.

What is the final value of j in this C program?

```
#include <stdio.h>


int increment (int x) {
    static int count = 0;
    count += x;
    return (count);

}


int main () {

    int i,j;

    for ( i=1; i<=4; i++ ) {
        j = increment(i);
    }

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

    return 0;

}
```

only initialized once

i=0	i=1	i=2	i=3	i=4
count=0	count=1	count=1	count=3	count=6
count=0	count=1	count=3	count=6	count=10
return 0	return 1	return=3	return=6	return=10

i=0 i=1 i=2 i=3 i=4 i=5 ← false, loop ends

j=0 j=1 j=3 j=6 j=10

j=4

Prints 10

(8)

What is the result of this program?

```
#include <stdio.h>

int main() {

  char* pointer = NULL;

  printf("%p\n", (void*) &pointer);

  return 0;

}
```

$\overbrace{\texttt{NULL}}^{\text{0 in C}}$

→ creates a pointer that points to 0x0

$\overbrace{\texttt{\&pointer}}^{\text{dereferencing the pointer}}$

↓ will print the pointer to a pointer

will print some address value

(2)

```
#include <stdio.h>
int main()
{
        int x, y;
        int* p = &x;
        int *q = &y;
        *p = 15;
        *q = *p + 10;
        printf("%d %d\n", x,y);
        return 0;
}
```

ptr = address

*p = 15; → value of x = 15

*q = *p + 10; → value of y = 25

printf("%d %d\n", x,y); → 15, 25

What would be the output of above code snippet?

(6)

What is the output of C program?

```
#include <stdio.h>

int main() {

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

    int b[4]= {5,6,7,8};

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

}
```

→ This above would not be a legal declaration

both are legal initializations

1 5

prints: 1,5

(7)

What is the result of this program:

```
#include <stdio.h>

int main() {

  int x[1] = {10};

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

  return 0;

}
```

int x[1] = {10}; ← legal

printf("%d\n", x[1]); ← illegal, array is of size 1, max index is 0.

Will result in stack buffer overflow