On the bubble sheet, please write your full name (as it would appear in D2L). You do not need to fill out any of the other fields (ID, section, etc.). Answer the following questions on the bubble sheet:

1. Which Linux command prints the contents of the current directory?

   A. `cd`
   B. `mv`
   C. `ls`
   D. `ssh`
   E. `cp`

2. Consider the following code snippet:

```
int num;
printf("Enter a number: ");
scanf("%d", num);
```

Will this code successfully assign an integer to the variable `num`?

   A. Yes
   B. No

3. Consider the following code snippet:

```
char string[20];
printf("Enter your name: ");
scanf("%s", string);
```

Will this code successfully assign a string to the variable `string`?

   A. Yes
   B. No

4. What should be filled in for the blank (`____`) so that the following prints "The temperature is 98.6"?

```
double temp = 98.6;
printf("The temperature is ____\n", temp);
```

   A. `temp`
   B. `%.2d`
   C. `%.1d`
   D. `%.2f`
   E. `%.1f`

5. What is the return type of the function `main`?

   A. `int`
   B. `double`
   C. `void`
   D. `char`

6. Suppose that you would like to save the current state of your project that is being tracked by `git`. Which of the following is the correct order that you would run the two commands?

   A. `git add` then `git commit`
   B. `git commit` then `git add`

7. Suppose the function `sum` returns a `double` and takes in two `double`s. Which of the following would be a valid function prototype for `sum`?

   A. `int sum(int, int);`

B. `double sum(param1, param2);`
C. `sum(double, double);`
D. `double sum(double, double);`
E. `#define sum;`

8. How do you start a comment in C?

A. `#`
B. `"`
C. `//`
D. `*`

9. Suppose your C source code is saved in a file called my `program.c`, and you compile it using the command `gcc -o my_program -Wall my program.c`. What is the `-o my_program` part of the command doing?

A. Saving the compiled file to `my_program`
B. Indicating where the source code is located
C. Running the preprocessor
D. Telling the compiler to check for warnings
E. Telling the compiler to check for syntax errors

10. What is the logical or operator in C?

A. `or`
B. `&`
C. `&&`
D. `|`
E. `||`

11. Suppose that your code contains the following snippet:

```
int num, status;
printf("Enter a number: ");
status = scanf("%d", &num);
```

Assume that this code is run and the user enters 10 like so:

```
Enter a number: 10
```

What is the value of `status`?

A. 10
B. 0
C. 1
D. 2
E. This code would not compile

12. What is the value of the variable `payout` after the following code snippet is run?

```
char flip = 'h';
int payout = flip == 't' ? 100 : -100;
```

A. -100
B. 0
C. 1
D. 100
E. This code would not compile

13. Suppose I have written a "Hello, world!" C program that just prints "Hello, world!" and I have compiled it into an executable called `hello`. What Linux command could I use to run `hello` and redirect the output to the file `out.txt`?

A. `hello out.txt`
B. `./hello`
C. `./hello > out.txt`
D. `./hello < out.txt`
E. `printf("Hello, world!", "out.txt")`

14. Suppose I have written a C program that reads in numbers from the console with `scanf` and I have compiled it into an executable called `sum`. What Linux command could I use to run `sum` and redirect the input from the console to the file `in.txt`?

A. `sum in.txt`
B. `./sum`
C. `./sum > in.txt`
D. `/.sum < in.txt`
E. `./sum 1 2 3`

15. Consider the following code snippet:

```
int num = 5;
do {
    printf("Num is less than 4\n");
} while (num < 4);
```

How many times does `Num is less than 4` print when this code is run?

A. 0
B. 1
C. 2
D. 3
E. This code would not compile

16. Can a function in C return an array?

A. Yes
B. No

17. After the following code is run, what is the value of `n`?

```
int n;
```

A. 0
B. -1
C. `NULL`
D. This code would not compile
E. We don't know

18. After the following code is run, what is the value of `j`?

```
int i = 2;
int j;
j = i++;
```

A. 0
B. 1
C. 2

D. 3
E. This code would not compile

19. After the following code is run, what is the value of `j`?

```
int i = 2;
int j;
j = ++i;
```

A. 0
B. 1
C. 2
D. 3
E. This code would not compile

20. Suppose I declare an array like so:

```
double nums[99];
```

How many slots for doubles does `nums` have?

A. 0
B. 99
C. 100
D. 101
E. we don't know

21. When the following C code is run, what is printed?

```
#include <stdio.h>
void func1(int x) {
    x++;
}
int main(void) {
    int x = 100;
    func1(x);
    printf("x is %d\n", x);
    return(0);
}
```

A. `x is 100`
B. `x is 101`

22. When the following C code is run, what is printed?

```
#include <stdio.h>
void func1(int x[]) {
    x[0]++;
}
int main(void) {
    int x[] = {100}; // this creates an array of size 1 with 100 in the first slot
    func1(x);
    printf("x[0] is %d\n", x[0]);
    return(0);
}
```

A. `x[0] is 100`
B. `x[0] is 101`

23. Suppose a variable `x` is declared like so:

```
int x = 5;
```

What C command would produce a pointer to `x`?

A. `ptr(x)`
B. `*x`
C. `&x`
D. `x*`
E. It's not possible to create a pointer to `x` because it was not declared as a pointer

24. Suppose the file `input.txt` has 5 doubles. If I write my C program `read.c` to read in the doubles using `scanf` and compile my program into an executable called `read`, which of the following would allow me to read in the doubles from `input.txt`?

A. `./read(input.txt)`
B. `./read < input.txt`
C. `./read input.txt`
D. `./input.txt`
E. `./read`

25. Suppose the file `input.txt` has 5 doubles. If I write my C program `read.c` to read in the doubles using `fscanf` and compile my program into an executable called `read`, which of the following would allow me to read in the doubles from `input.txt`?

A. `./read(input.txt)`
B. `./read < input.txt`
C. `./read input.txt`
D. `./input.txt`
E. `./read`

26. Consider the following variable declaration:

```
int *num;
```

What is the data type of `num`?

A. int
B. pointer to an int
C. it hasn't been initialized so it doesn't have a data type yet

27. Suppose the function `add` is defined as:

```
void add(int a, int b, int* result) {
    *result = a + b;
}
```

And suppose we have variables `x`, `y`, and `sum` as follows:

```
int x = 10, y = -2, sum;
```

How can we call the function `add` so that the variable `sum` will hold the value 8 after it is run?

A. `add(*x, *y, *sum)`
B. `result = add(x, y)`
C. `add(x, y, *sum)`
D. `add(x, y, &sum)`
E. `add(&x, &y, &sum)`

28. What is `sizeof(int)` on our server?

A. 1
B. 2
C. 4
D. 8
E. 12

29. What causes a segmentation fault?

A. Forgetting a semicolon at the end of a C command
B. Not freeing memory
C. Using heap memory instead of stack memory
D. Accessing memory in a way that is not allowed
E. Passing by value

30. After the following code is run,

```
int *b;
int n = 5;
b = &n;
```

which of the following changes the value of `n`?

A. `*b = 6`
B. `&b = 6`
C. `b = 6`

31. If `a` and `b` are strings, how can we determine whether `a` comes first alphabetically or not?

A. `a < b`
B. `strcmp(a, b) < 0`
C. `strcmp(*a, *b) < 0`
D. `*a < *b`
E. `strcmp(&a, &b) < 0`

32. If `x` is defined as follows, what is the result of `strlen(x)`?

```
char x[10] = "hi";
```

A. 10
B. 3
C. 2

33. Using the same `x` as in the problem before, what is the result of `sizeof(x)`?

A. 80
B. 40
C. 10
D. 3
E. 2

34. When the following code is run, what prints?

```
char str[20] = "Adams, John Quincy";
strtok(str, ", ");
printf("%s\n", str);
```

A. `Adams, John Quincy`
B. `John`
C. `Quincy`
D. `Adams`

E. `Adams  John Quincy`

35. Suppose we have defined a struct `City` like so:

```
typdef struct {
    char name[20];
    int pop;
    int elevation;
} City;
```

and then declared and initialized a `City`:

```
City c = {"Bozeman", 54539, 4817};
```

How do we change `c`'s `pop` field?

A. `City.pop = 55345;`
B. `c.pop = 55345;`
C. `strcpy(pop, 55345);`
D. `pop = 55345;`
E. `pop->55345`

36. Are structs passed by value or by reference?

A. Value
B. Reference

37. Can a function return a struct?

A. Yes
B. No

38. Suppose the function `update_pop` is defined as follows:

```
void update_pop(City *c) {
    printf("Enter a new population: ");
    scanf("%d", &c->pop);
}
```

How would we call `update_pop` if we have a `City` stored in the variable `bzn`?

A. `update_pop(bzn);`
B. `update_pop(*bzn);`
C. `update_pop(&bzn);`
D. `update_pop(bzn.pop);`
E. `bzn = update_pop(*bzn);`

39. In the function `update_pop` above, what is the data type of the variable `c`?

A. A pointer to a `City`
B. A pointer to an `int`
C. A `City`
D. An `int`
E. A struct

40. In the function `update_pop` above, what is the data type of the expression `&c->pop`?

A. A pointer to a `City`
B. A pointer to an `int`
C. A `City`
D. An `int`

E. A struct

41. Suppose we want to print out the population of the variable `bzn`, which is a `City` struct as above. Which of the following would do that?

A. `printf("%d\n", bzn->pop)`
B. `printf("%d\n", bzn.pop)`
C. `printf("%d\n", bzn(pop))`
D. `printf("%d\n", &bzn->pop)`

42. When running `gcc`, what does the `-c` flag do?

A. Produces object files instead of executable files.
B. Runs the linker.
C. Specifies a name for the executable.
D. Compiles C code instead of C++ code.

43. Which of the following rules could we put in a Makefile so that `student.o` is recompiled whenever `student.h` or `student.c` is changed?

A.

```
student.o: student.h student.c
        gcc student.c student.h -Wall
```

B.

```
student.o:
        gcc student.c student.h -Wall
```

C.

```
student.o: student.h student.c
        gcc -c student.c -Wall
```

D.

```
student.o: student.h student.c
        gcc -o student.c main.c -Wall
```

44. Header (`.h`) files are never compiled.

A. True
B. False

45. Suppose that a `planet_t` is a struct with a field called `name`, and that `p` is a pointer to a `planet_t`. What is `p->name` equivalent to?

A. `p.name`
B. `strcpy(p.name, x)`
C. `(*p).name`
D. `&p.name`

46. Suppose that we compile C source code defining main as follows

```
int main(int argc, char* argv[])
```

into an executable called `exe`. If we run `exe` with `./exe 1 2 3`, what is `argv[0]`?

A. `1 2 3`
B. `1`
C. `2`

D. `3`
E. `./exe`

47. Running `exe` as in the previous problem, what is the data type of `argv[2]`?

A. Null pointer
B. String
C. Int
D. Char
E. Double

48. Suppose I run the following:

```
char names[][10] = {"Bob", "Sally", "Mary"};
```

What is `sizeof(names)`? Remember that `char`s take up 1 byte.

A. 3
B. 12
C. 15
D. 18
E. 30

49. Suppose that `int_ptrs` is an array of 10 pointers to ints. For example, it may have been declared using `int* int_ptrs[10];`. How do we get the int that `int_ptrs[0]` points to?

A. `*int_ptrs[0]`
B. `&int_ptrs[0]`
C. `int_ptrs[0]`
D. `int_ptrs[0]*`

50. What do we input to a call to `malloc`?

A. A pointer to the data we want to store on the heap
B. The data that we want to store on the heap
C. The size in bytes of the data that we want to store on the heap
D. A `NULL` pointer

51. When we run `valgrind` to check for memory issues, do we run it on the C source code or the compiled C program?

A. Source code
B. Compiled program

52. Suppose I want to allocate space for 20 `City` structs on the heap. Which of the following would do that?

A. `City *cityarr = calloc(20, sizeof(City));`
B. `City cityarr[20] = calloc(20, sizeof(City));`
C. `City cityarr[20];`
D. `City *cityarr = (City*) calloc(City);`
E. `City &cityarr = (City*) calloc(City);`

53. What is the bitwise not operator in C? (That is, the operator that flips individual bits from 0 to 1 or vice versa.)

A. `!`
B. `&`
C. `|`
D. `!!`

E. ~

54. What is 15 in binary?

A. 1111
B. 1000
C. 0101
D. 1001
E. 1110

55. What is 15 in hexadecimal?

A. 10
B. 15
C. f
D. a
E. x