```
1.  Array
2.  Array[3]
3.  &Array[3]
4.  *(&Array[3])
5.  int Array[10];
6.  int x;
7.  int* px;
8.  &x
9.  *px
10. x
```

Illustration 1: Declarations and References

1. Illustration 1: Which line declares an array?
2. Illustration 1: Which line declares an integer?
3. Illustration 1: Which line declares an integer pointer?
4. Illustration 1: Which line gives the value of the fourth element of an array?
5. Illustration 1: Which line gives the address of the third subscripted value of an array?
6. Illustration 1: Which line specifies the value of an integer?
7. Illustration 1: Which line gives the pointer to the first element of an array?
8. Illustration 1: Which line gives the contents of the address of the fourth element of an array?
9. Illustration 1: Which line gives the address of an integer?
10. Which line dereferences (gives the contents) of an address given by an integer pointer?

```
1. int x;
2. int cherry[10];
3. foo(x);
4. foo(&x);
5. foo(cherry[3]);
6. foo(cherry, 3);
7. foo(&cherry[3]);
```

Illustration 2: Function calls

11. Illustration 2: Which line declares an integer?
12. Illustration 2: Which line declares an array with 10 elements?
13. Illustration 2: Which line calls foo with a copy of an integer?
14. Illustration 2: Which line calls foo with the address of an integer?
15. Illustration 2: Which line calls foo with the address of the first element of an array?
16. Illustration 2: Which line calls foo with the value of an element of an array?
17. Illustration 2: Which line calls foo with the address of an element of an array which is not the first element?

```
1. int foo( int boot[5] );
2. int foo( int* boot );
3. int foo( int boot[] );
4. int foo ( int x );
5. int foo( int *x );
6. int foo( int tire[3][5] );
```

Illustration 3: Function prototypes

18. Illustration 3: What makes the lines a function prototype?
19. Illustration 3: Which line declares a single dimension integer array with an unnecessary dimension value?
20. Illustration 3: Which line declares a multi-dimension array?
21. Illustration 3: Which line declares an integer array without including dimension information?
22. Illustration 3: Which line declares that the function is receiving a copy of a value?
23. Illustration 3: Does line 1 declare an integer pointer?
24. Illustration 3: Does line 2 declare an integer pointer?
25. Illustration 3: Does line 3 declare an integer pointer?
26. Illustration 3: Can the integer pointer from line 2 be used with subscripts?
27. Illustration 3: Can the integer pointer from line 3 be used

with subscripts?

28. Illustration 3: Which line above declares an integer value?

29. Illustration 3: Could line 5 be used to declare an array?

30. Illustration 3: Could line 5 be used to declare the pointer to a single integer value?

31. Illustration 3: Which line is most useful to declare a reference to (the pointer to) a specific element of an integer array (such as &donut[4]) ?

32. Illustration 3: Which line is most useful to declare the value of a specific element of an integer array (such as donut[4]) ?

```
1. int foo( tube[] )
2. int foo( int x )
3. int foo( int* px )
4. int foo( int* tube, size)
```

Illustration 4: Function definintions

33. Illustration 4: Which function definition lines can be called with the name of an array?

34. Illustration 4: Which function definition lines can be called with a copy of the original value?

35. Illustration 4: Which function definition can be called with an array value (like foo(array[3]);)?

36. Illustration 4: Which function definition can include the dimension property of an array?

37. Illustration 4: Which function definition can be called with the address of an integer value?

```
1. int boot [10];
2. int x;
3. y = foo(x);
4. y = foo(boot[5]);
5. y = foo(&x);
6. y = foo(boot);
7. y= foo(boot, 10);
```

Illustration 5: Function calls

38. Illustration 5: Which line calls a function with the name of an array?

39. Illustration 5: Which line calls a function with a pointer to an array?

40. Illustration 5: Which line calls a function with dimension property of an array passed as an integer?

41. Illustration 5: Which line calls a function with a copy of an array element?

42. Illustration 5: Which line calls a function with the address of an integer?

43. Illustration 5: Which line calls a function with the copy of an integer value?

44. String is a data type in C? (TRUE or FALSE)

45. Strings are letters placed in character arrays ending with a null character - (TRUE or FALSE)

46. Illustration 6: Which line declares a string (char array) with maximum 80 characters, initialized to a string?

47. Illustration 6: Which line just creates an array useful for 80 characters?

```
1.char sta[80];
2.char text[80]=”This is the value”;
3.char* ptext=”Sample string”;
4.char output[120];
5.puts(text);
6.printf(“%s”,text);
```

Illustration 6:

48. Consider Illustration 6 line 5 outputs string array **'text'** with a new line included? (TRUE or FALSE)

49. Consider Illustration 6 line 5, what does the calling argument supply to the function?
(VALUE, or POINTER)

50. Consider Illustration 6 which line uses the string format specifier?

51. Consider Illustration 6 line 6, what does the second calling argument supply to the function?
String (VALUE, or POINTER)

52. Consider Illustration 6 line 6, what does the first calling argument supply to the function?
String (VALUE, or POINTER)

53. Consider Illustration 6 Which line creates a string of letters and stores them in array **'text'**?

54. Consider Illustration 6 line 3 creates a string constant and stores what in ptext?
    (VALUE, or POINTER)

55. Illustration 6: Which string library function can move **'text'** to **'output'**?

56. Illustration 6: Which string library function can add **'ptext'** to **'output'**?

57. Illustration 6: Which string library function can add "this is more text" to **'output'**?

58. Illustration 6: Which string library function can tell how many characters are in **'text'**?