Question 1 **1 / 1 point**

The pool of memory used for dynamically allocated data is sometimes called the _____.

- global storage area
- ✔ heap
- activation record
- run-time stack

Question 2 **1 / 1 point**

Which of the situations might cause run-time stack overflow?

- The main function creates a huge array of capacity in the millions
- There is a very large number of user-defined functions, each calling several other functions
- ✔ Both situations might cause stack overflow
- Neither situation could cause stack overflow

Question 3 **1 / 1 point**

Which line of code creates a dynamic array?

- ✔ int * a = new int [4];
- int a[4];

- int & a = int [4];
- int a [ ] = { 10, 20, 30, 40 };

Question 4 **1 / 1 point**

The code below generates a syntax error as variable array cannot be altered.

```
int * array = new int [4];
int k = 10;
array = &k;
```

- True
- ✓ False

Question 5 **1 / 1 point**

Which statement is not true about the code? Assume an integer uses 4 bytes.

```
int * values = new int [2];
value[0] = 15;
values[1] = 25;
delete [ ] values;
values = NULL;
```

- The dynamic array pointed to by **values** was established with 8 bytes
- After the code completes, attempting to dereference pointer **values** is illegal
- ✓ The memory used by the dynamic array is available for the rest of the time the program is executing via pointer variable **values**
- After the code completes, memory used by the dynamic array has been released

Question 6 **0 / 1 point**

The code below creates _____.

```
int * p1 = new int;
*p1 = 15;
int * p2 = p1;
delete p1;
```

- a memory leak
- a dangling pointer
- both a memory leak and a dangling pointer
- neither a memory leak nor a dangling pointer

Question 7 **1 / 1 point**

If a user-defined function deletes a dynamic array and then dereferences the pointer that stores its base address, a _____ problem occurs.

- memory leak
- dangling pointer
- stack overflow
- garbage collector

Question 8 **1 / 1 point**

What displays when this code executes?

```
char [ ] word = "balloon";
cout << word;
```

- b

✔ balloon

balloon0

the base address of the array

Question 9 **1 / 1 point**

A C string is stored as a nul-terminated char array.

✔ True

False

Question 10 **1 / 1 point**

The code below can be used to resize array **a** so that it is twice as big.

```
int a[5];
a = new int [10];
```

True

✔ False

Question 11 **0 / 1 point**

Rectangle is a user-defined class with a public method named **area**. Which code segment correctly calls the area method?

```
Rectangle * r = new Rectangle;
cout << r.area( );
```

➡
```
Rectangle * r = new Rectangle;
cout << r->area( );
```

```
Rectangle r = new Rectangle;
cout << r->area( );
```

✖ **All choices are correct**

Question 12 **1 / 1 point**

Code uses the **find** function in the **algorithm** library. What displays when the code executes?

```
int a[4] = { 10, 20, 30, 40 };
int * location = find (a, a+4, 30);
cout << location;
```

- 30
- 40
- ✓ the address where 30 is stored
- the address at the end of the array

Question 13 **1 / 1 point**

Code uses the **find** function in the **algorithm** library. What condition should be used in the if statement to determine if the search was successful?

```
int a[4] = { 10, 20, 30, 40 };
int * location = find (a, a+4, 35);
if ( _____ )
      cout << "failed search";
```

- location == a
- location == -1
- location == NULL
- ✓ location == a+4

Question 14 **1 / 1 point**

Use the _____ function in the **algorithm** library to perform a fast search on a

sorted array.

- ✔ binary_search
- sort
- swap
- find

Question 15 **1 / 1 point**

Code needs to exchange the values in the first two indexes of the array. Which function call is correct using the **swap** function in the **algorithm** library?

int a[4] = { 10, 20, 30, 40 };

- swap (&a, &(a+1));
- swap (10, 20);
- ✔ swap (a[0], a[1]);
- swap (a, a+1);

Question 16 **1 / 1 point**

The **fill** function in the **algorithm** library can be used with different types of arrays, as shown in the code.

```
int a[4] = { 0 };
string b[10] = { " " };
fill (a, a+4, 15);
fill (b, b+10, "hello");
```

- ✔ True

- False

Question 17 **1 / 1 point**

What is the effect of the code?

```
int a[4] = { 10, 20, 30, 40 };
int * p1 = a;
int * p2 = &a[0];
if (p1 == p2)   cout << "aaa";
else if (p1 < p2)   cout << "bbb";
else cout << "ccc";
```

- ✔ It displays: aaa
- It displays: bbb
- It displays: ccc
- It generates a syntax error

Question 18 **1 / 1 point**

Use of the C++ string class is generally less prone to error than use of the C string library as the public methods automatically resize the underlying array as needed.

- ✔ True
- False

Question 19 **1 / 1 point**

Which for loop correctly iterates over the first 2 elements of the array?

```
int * a = new int [4];
a[0] = 20;   a[1] = 25;  a[2] = 30;  a[3] = 35;
```

-
```
for (int * ptr = a; ptr < a + 2; ptr++)
        cout << *ptr << endl;
```

```
for (int k = 0;  k < 2; k++)
        cout << a[k] << endl;
```

✔ Both solutions are correct

Neither solution is correct

Question 20 **1 / 1 point**

Continual omission of the **delete [ ]** operator using large dynamic arrays in a program could eventually result in a heap overflow error.

✔ True

False