Q-21

Due No due date **Points** 15 **Questions** 20 **Time Limit** 30 Minutes **Allowed Attempts** Unlimited

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	20 minutes	14 out of 15

Correct answers are hidden.

Submitted Jul 21 at 10:09pm

Question 1

1 / 1 pts

Below is ***insert()***, a template function that works with a ***partially-filled array***. The function inserts the argument ***e*** into the array, in sorted order. The function returns ***true*** if it succeeds, ***false*** otherwise. The function contains an error; what is the error?

```
template <typename T>
bool insert(T* a, size_t& size, size_t MAX, T e)
{
    if (size < MAX) return false;
    size_t i = 0;
    while (i < size)
    {
        if (a[i] > e) break;
        i++;
    }
    for (j = size; j > i; j--)
        a[j] = a[j - 1];
    a[i] = e;
    size++;

    return true;
}
```

- When a value is inserted, it erases one of the existing values
- The second loop should start at `i` and go up to `size`
- None of these
- (selected) If there is room to insert, the function returns `false` instead of `true`. It should say `if (size == MAX)`
- The value is inserted into the wrong position

Question 2

1 / 1 pts

Below is ***startsWith()***, a template function that works with two ***partially-filled arrays***. The function returns ***true*** if the array ***a*** "starts with" the same elements as the array ***b***, ***false*** otherwise. The function contains an error; what is the error?

```
template <typename T>
bool startsWith(const T* a, size_t sizeA, const T* b, size_t sizeB)
{
    if (sizeA > sizeB) return false;
    for (size_t i = 0; i < sizeB; i++)
        if (a[i] != b[i]) return false;
    return true;
}
```

- `sizeA` and `sizeB` should both be passed by reference
- The condition `a[i] != b[i]` should be `b[i] == a[i]`
- None of these
- The condition `i < sizeB` should be `i <= sizeB`
- (selected) The condition `(sizeA > sizeB)` should be `(sizeB > sizeA)`

Incorrect

Question 3

0 / 1 pts

Below is ***insert()***, a template function that works with a ***partially-filled array***. The function inserts the argument ***e*** into the array, in sorted order. The function returns ***true*** if it succeeds, ***false*** otherwise. The function contains an error; what is the error?

```
template <typename T>
bool insert(T* a, size_t& size, size_t MAX, T e)
{
    if (size >= MAX) return false;
    size_t i = 0;
    while (i < size)
    {
        if (a[i] > e) break;
        i++;
    }
    for (j = size; j > i; j--)
        a[j] = a[j - 1];
```

```
    a[i] = e;

    return true;
}
```

- ○ The value is inserted into the wrong position
- ◉ The function writes over memory outside the array when it should not
- ○ Every time the function is called, an array element is "lost"
- ○ The second loop should start at `i` and go up to `size`
- ○ None of these

Question 4

1 / 1 pts

Below is a declaration for a ***partially-filled array***. What is the correct prototype for a function ***add()*** that appends a new element to the end of the array and returns ***true*** if successful?

```
const size_t MAX = 100;
double nums[MAX];
size_t size = 0;
```

- ○ `bool add(double a[], size_t size, size_t MAX, double e);`
- ◉ `bool add(double a[], size_t& size, size_t MAX, double e);`
- ○ None of these
- ○ `bool add(double a[], size_t MAX, double e);`
- ○ `bool add(double a[], size_t& size, double e);`

Question 5

1 / 1 pts

Assume you have a ***partially filled array a***, with variables ***size*** and ***MAX*** (capacity). To append ***value*** to the array, which of these assignments is correct?

- ◉ `a[size] = value;`
- ○ `a[MAX - 1] = value;`
- ○ None of these
- ○ `a[size + 1] = value;`
- ○ `a[size - 1] = value;`

Question 6

0.5 / 0.5 pts

When comparing two partially-filled arrays for equality, both arrays should not be declared `const`.

- ○ True
- ◉ False

Question 7

0.5 / 0.5 pts

In a partially-filled array, all of the elements contain meaningful values

- ○ True
- ◉ False

Question 8

0.5 / 0.5 pts

In a partially-filled array ***capacity*** represents the number of elements that are in use.

- ○ True
- ◉ False

Question 9

0.5 / 0.5 pts

When comparing two partially-filled arrays for equality, both arrays should be declared `const`.

- ◉ True

- False

Question 10

0.5 / 0.5 pts

When searching for the index of a particular value in a partially-filled array, the array should be declared `const`.

- (•) True
- ( ) False

Question 11

1 / 1 pts

Which statement displays the value 24 from the 2D array initialized here?

```
int a[2][3] =
{
   { 13, 23, 33 },
   { 14, 24, 34 }
};
```

- (•) `cout << a[1][1];`
- ( ) `cout << a[2][1];`
- ( ) `cout << a[2][2];`
- ( ) `cout << a[2][2];`
- ( ) `cout << a[1][2];`
- ( ) None of these

Question 12

1 / 1 pts

What prints?

```
int x = 0;
int a[2][3] = {{1, 2, 3}, {4, 5, 6}};
for (auto r : a)
    for (auto c : r) x++;
cout << x << endl;
```

- ( ) 2
- ( ) Undefined (out of bounds)
- (•) Illegal; will not compile
- ( ) 3
- ( ) 6

Question 13

1 / 1 pts

What prints?

```
int a[3][2] = {{3,2,3}};
cout << a[0][0] << a[1][0] << endl;
```

- ( ) 00
- ( ) 30
- ( ) 31
- ( ) 33
- (•) Code does not compile

Question 14

1 / 1 pts

What prints?

```
int cnt = 0, a[4][5];
for (int i = 0; i < 5; i++)
   for (int j = 0; j < 4; j++)
      a[j][i] = cnt++;
cout << a[2][3] << endl;
```

- (•) 14
- ( ) 9

- 11
- 19
- 8

Question 15

1 / 1 pts

What prints?

```
int a[5][3] = {
    { 1,  2,  3},
    { 4,  5,  6},
    { 7,  8,  9},
    {10, 11, 12},
    {13, 14, 15}
};

int *p = &a[0][0];
cout << p[10] << endl;
```

- 11
- Undefined (out of bounds)
- An address
- 10
- Illegal; will not compile

Question 16

0.5 / 0.5 pts

When initializing a 2D, each column must have its own set of braces.

- True
- False

Question 17

0.5 / 0.5 pts

In a 2D array the first subscript represents the columns and the second the rows.

- True
- False

Question 18

0.5 / 0.5 pts

You can pass the 2D array ***`int a[3][3]`*** to the function ***`f(int *a, size_t r, size_t c)`*** by calling ***`f(&a[0][0], 3, 3)`***.

- True
- False

Question 19

0.5 / 0.5 pts

When passing a 2D array to a function, the array parameter must explicitly list the size for all dimensions except for the last, like: ***`void f(int a[3][], size_t n);`***

- True
- False

Question 20

0.5 / 0.5 pts

You can pass the 2D array ***`int a[3][3]`*** to the function ***`f(int *a, size_t r, size_t c)`*** by calling ***`f(a, 3, 3)`***.

- True
- False