Instructions: This homework assignment focuses on basic facts about using pointers in C++. Submit your answers via the Curator as Quiz: Pointers.

For the next three questions, consider the following short program:

```
#include <iostream>
#include <new>
using namespace std;

int main() {

   char* pC = NULL;    // Line 1
   pC = new char;      //      2
   *pC = 'x';          //      3

   return 0;
}
```

1. Which of the following best describes the situation after Line 1 has been executed?

   1) pC does not have a target, and pC does not have a known value.
   2) pC does not have a target, but pC does have a known value.
   3) pC has a target, but its target does not have a known value.
   4) pC has a target, and its target does have a known value.
   5) None of these

2. Which of the following best describes the situation after Line 2 has been executed?

   1) pC does not have a target, and pC does not have a known value.
   2) pC does not have a target, but pC does have a known value.
   3) pC has a target, but its target does not have a known value.
   4) pC has a target, and its target does have a known value.
   5) None of these

3. Which of the following best describes the situation after Line 3 has been executed?

   1) pC does not have a target, and pC does not have a known value.
   2) pC does not have a target, but pC does have a known value.
   3) pC has a target, but its target does not have a known value.
   4) pC has a target, and its target does have a known value.
   5) None of these

For the next two questions, consider the following short program:

```
#include <iostream>
using namespace std;
int* F();
int* G();

int main() {
   int* f = F();
   int* g = G();

   cout << *f << endl;
   return 0;
}
```

```
int* F() {
   int i = 42;
   return &i;
}

int* G() {
   int  j = 25;
   return &j;
}
```

4. Upon executing the program, the author is surprised by the value that is printed. What value was printed?

   1) 42 2) 25 3) Neither of these

5. The surprise resulted from a classic logic error in the given code. What is that error?

   1) Using a pointer as a return value from a function
   2) Returning a pointer value that has not been set
   3) Returning a pointer whose target is a local variable within the called function
   4) None of these

6. Would the result be different if the call to the function `G()` was omitted, and if so, why?

   1) No
   2) Yes, because `G()` returns an incorrect value.
   3) Yes, because the execution of the call changes the value of the pointer `f`.
   4) Yes, because the execution of the call changes the value of the target of the pointer `f`.
   5) Yes, but for a reason that is not listed.

For questions 7 through 9, assume that `P` and `Q` are pointers of the same type, and that each has been assigned a value.

7. What comparison would reliably decide if `P` and `Q` have targets with the same value?

   1) `P == Q`
   2) `*P == *Q`
   3) `&P == &Q`
   4) All of them
   5) 1 and 2 only
   6) 1 and 3 only
   7) 2 and 3 only
   8) None of these

8. What comparison would reliably decide if `P` and `Q` have the same target?

   1) `P == Q`
   2) `*P == *Q`
   3) `&P == &Q`
   4) All of them
   5) 1 and 2 only
   6) 1 and 3 only
   7) 2 and 3 only
   8) None of these

9. What comparison would reliably decide if `P` and `Q` have the same value?

   1) `P == Q`
   2) `*P == *Q`
   3) `&P == &Q`
   4) All of them
   5) 1 and 2 only
   6) 1 and 3 only
   7) 2 and 3 only
   8) None of these

Consider the main function below:

```
int main() {

   // Find number of data values:
   ifstream In("Temperature.data");
   if ( In.fail() ) {
      cout << "Input file not found: " << endl;
      return 1;
   }

   int numReadings;
   In.ignore(INT_MAX, ':');
   In >> numReadings;

   // Create data array:
   int *Temperature = NULL;
   Temperature = new int[numReadings];

   // Initialize data array:
   initArray(Temperature, numReadings, INT_MIN);

   // Acquire data values:
   if ( !acquireData(Temperature, numReadings, In) ) {
      cout << "Incorrect number of data values in input file." << endl;
      return 1;
   }

   // Average data values:
   double averageTemperature = Average(Temperature, numReadings);

   cout << fixed << showpoint;
   cout << "Average of " << numReadings << " temperatures was "
        << setprecision(1) << averageTemperature << endl;


   cout << "Maximum temperature was "
        << maxEntry(Temperature, numReadings) << endl;

   In.close();
   return 0;
}
```

The program is to read data from a file like this:

```
# of data values: 10
  1:  87
  2:  84
  3:  79
  4:  76
  5:  72
  6:  69
  7:  68
  8:  66
  9:  65
 10:  63
```

For the next two questions, consider implementing the function to read the temperature data:

```
bool acquireData(int A[], int numValues, ifstream& In) {      // Line 1

   int Pos;                                                   //      2
   for (Pos = 0; In && Pos < numValues; Pos++) {              //      3
                                                              //      4
      In >> A[Pos];                                           //      5
   }

   return ( Pos == numValues );                               //      6
}
```

10. How should the blank in Line 4 be filled?

1) `In >> ':';`
2) `In.ignore(INT_MAX, ':');`
3) `In.ignore(INT_MAX);`
4) None of these

11. Given the function parameter types above, is the first actual parameter used in the call in `main()` legal?

1) Yes
2) No
3) Not enough information

For the next four questions, consider implementing the function to calculate the average temperature. The `for` loop is required to be implemented using pointers to access elements rather than direct array indexing.

```
double Average(const int* const Data, int Sz) {               // Line 1

   if ( Data == NULL || Sz <= 0 ) return 0.0;                 //      2

   int Sum = 0;                                               //      3
   const int *Current = Data;                                 //      4
   for (int Pos = 0; Pos < Sz; Pos++) {                       //      5
      Sum = Sum + _________;                                  //      6
      ________________                                        //      7
   }
   return ( double(Sum) / Sz );                               //      8
}
```

12. What is the effect of the double use of `const` in Line 1?

1) The first prevents the pointer `Data` from being modified.
2) The first prevents the target of the pointer `Data` from being modified.
3) The second prevents the pointer `Data` from being modified.
4) The second prevents the target of the pointer `Data` from being modified.
5) 1 and 3 only
6) 1 and 4 only
7) 2 and 3 only
8) 2 and 4 only
9) None of these

13. `Current` is declared using `const` in Line 4. Does that conflict with the statement that must be given in Line 7?

1) Yes
2) No
3) Not enough information

14. How should the blank in Line 6 be filled?

1) `Current`
2) `&Current`
3) `*Current`
4) None of these

15. How should the blank in Line 7 be filled?

1) `Current++;`
2) `(*Current)++`
3) `Pos + 1`
4) It should be left blank.
5) None of these

For the next five questions, consider implementing the function to find the maximum temperature. The `for` loop is required to be implemented using pointers to access elements rather than direct array indexing.

```
int maxEntry(const int* const Data, int Sz) {                  // Line 1

   if ( Data == NULL || Sz <= 0 ) return INT_MIN;              //      2

   int Count = 0;                                              //      3

   // Set hiSoFar to point to the first array element:
   const int *hiSoFar = _________;                             //      4

   // Set Current to point to the second array element:
   const int *Current = _________;                             //      5

   for ( ; Count < Sz; _________________) {                    //      6

      if ( ____________________________ )                      //      7

         hiSoFar = Current;                                    //      8
   }

   return ( _______________ );                                 //      9
}
```

16. How should the blank in Line 4 be filled?

1) `&Data`
2) `*Data`
3) `Data`
4) `Data[0]`
5) `&Data[0]`
6) 3 or 5 only
7) 3 or 4 only
8) None of these

17. How should the blank in Line 5 be filled?

1) `hiSoFar`
2) `hiSoFar++`
3) `Data++`
4) `Data[1]`
5) `&Data[1]`
6) 2 or 4 only
7) 2 or 5 only
8) None of these

18. How should the blank in Line 6 be filled?

1) `Count++`
2) `Current++`
3) `Count++, Current++`
4) It should be left blank.
5) None of these

19. How should the blank in Line 7 be filled?

1) `*Current > *hiSoFar`
2) `Current > hiSoFar`
3) `&Current > &hiSoFar`
4) `*Current > *hiSoFar`
5) None of these

20. How should the blank in Line 9 be filled?

1) `hiSoFar`
2) `*hiSoFar`
3) `&hiSoFar`
4) It should be left blank.
5) None of these

For the next three questions, assume the following memory contents with the declaration: `int *A;`

	Address (hex)	Value (hex)
A	0012FED4	002F1090
	002F1090	0000002A
	0000002A	00140B1D

Choose from the following answers (formatting is unimportant):

1) `0012FED4`
2) `002F1090`
3) `0000002A`
4) `002F1090`
5) `0000002A`
6) `00140B1D`
7) None of these

21. What value would be written by the statement: `cout << hex << &A << endl;`

22. What value would be written by the statement: `cout << hex << A << endl;`

23. What value would be written by the statement: `cout << hex << *A << endl;`

Consider the following main function, which deals with a dynamically allocated array of pointers to string objects:

```
int main() {

   ifstream In("Text.data");

   int numNames;
   In >> numNames;
   In.ignore(INT_MAX, '\n');

   string** Name = new string*[numNames];
   // Set each array cell to NULL:
   initArray(Name, numNames);
   // Read strings from input file:
   acquireData(Name, numNames, In);
   // Display strings to verify input success:
   writeArray(Name, numNames);

   // Search for strings matching "bar":
   string Sought = "bar";
   int Matches = numMatches(Name, numNames, Sought);
   cout << "Number of matches for " << Sought
        << " is " << Matches << endl;

   In.close();
   return 0;
}
```

For the next two questions, consider the implementation of the input function:

```
bool acquireData(string** const A, int Sz, ifstream& In) {     // Line 1

   if ( A == NULL || Sz <= 0 ) return false;                    //      2

   string Temp;                                                 //      3
   for (int Pos = 0; In && Pos < Sz; Pos++) {                   //      4

      getline(In, Temp);                                        //      5
      if ( A[Pos] != NULL ) delete A[Pos];                      //      6
      A[Pos] = new string(Temp);                                //      7
   }
   return ( Pos == Sz );                                        //      8
}
```

24. What is the purpose of the statement in Line 6?

1) To prevent the program from writing data to an invalid address.
2) To prevent the program from reading data from an invalid address.
3) To prevent a memory leak; i.e., losing access to memory without deallocating it.
4) None of these

25. What is the purpose of the comparison used in Line 8?

1) To determine whether the expected number of data values was read.
2) To prevent the program from reading more than the specified number of data values.
3) To prevent the program from reading fewer than the specified number of data values.
4) None of these

For the next two questions, consider the implementation of the function to write the strings:

```
void writeArray(string** const A, int Sz) {     // Line 1

   if ( A == NULL || Sz <= 0 ) return;          //      2

   for (int Pos = 0; Pos < Sz; Pos++) {         //      3

      cout << setw(5) << Pos << ": ";           //      4

      if ( A[Pos] != NULL )                     //      5
         cout << *A[Pos] << endl;               //      6
      else
         cout << "null pointer" << endl;       //      7
   }
}
```

26. Consider the statement in Line 2. Assuming that `Sz` is <u>expected to be</u> the number of values stored in the array `A`, what is the purpose of the statement in Line 2?

    1) To prevent an access violation in Line 5 if the array hasn't been allocated.
    2) To prevent an access violation in Line 5 if the array doesn't contain any data values.
    3) 1 and 2
    4) None of these

27. What is the purpose of the test in Line 5?

    1) To prevent an access violation in Line 6 if the array hasn't been allocated.
    2) To prevent an access violation in Line 6 if the array doesn't contain any data values.
    3) 1 and 2
    4) None of these

For the next three questions, consider the function to search the array and count string matches:

```
int numMatches(string** const A, int Sz, string toMatch) {  // Line 1

   if ( A == NULL || Sz <= 0 ) return 0;                    //      2

   int Count = 0;                                           //      3

   for (int Pos = 0; Pos < Sz; Pos++) {                     //      4

      if ( ________________ && _______________ )            //      5
         Count++;                                           //      6
   }
   return Count;                                            //      7
}
```

28. In order to prevent an access violation, how should the <u>first</u> blank in Line 5 be filled?

    1) `A != NULL`
    2) `A[Pos] != NULL`
    3) `toMatch != ""`
    4) None is needed; it should be left blank and the `&&` should be removed.
    5) None of these

29. In order to correctly detect a match, how should the second blank in Line 5 be filled?

    1) `toMatch == *A[Pos]`
    2) `toMatch ==  A[Pos]`
    3) `&toMatch == *A[Pos]`
    4) 1 or 2 only
    5) 2 or 3 only
    6) 1 or 3 only
    7) None of these

30. Does the order of the two parts of the and-expression in Line 5 matter?

    1) No.
    2) Yes, if the order is reversed then valid matches may not be detected.
    3) Yes, if the order is reversed then matches may be reported when none occurred.
    4) Yes, if the order is reversed then access violations could occur if the array did not contain the expected number of strings.
    5) None of these