The diagram above is used for questions 1 – 4 below

1. Write 4 expressions that refer to the value of the element pointed to by letter “A” in the above diagram using array subscript/offset and pointer subscript/offset notation.

   \[ \text{Values}[4], \quad p[4], \quad *(\text{Values} + 4), \quad *(p + 4) \]

2. What would be the value of the expression \( p + 3 \) //pointer arithmetic!!!

   \[
   p = 1022 \\
   p + 3 = 1022 + (3 \times 4) = 1034
   \]

3. What would be the value of the expression \( *(\text{Values} + 5) \)

   \[ 9 \quad //\text{remember to start counting at 0…} \]

4. What would be the value of the expression \( p[2] + \text{Values}[5] \)

   \[ p[2] = 8, \quad \text{Values}[5] = 9, \quad 9 + 8 = 17 \]

5. List the C++ operators permitted for pointer arithmetic?

   \[ +, - , ++ , -- , += , -= \]

6. Write C++ statement(s) to: dynamically allocate memory for 100 integers using a pointer variable named ptr.

   \[ \text{int} \ * \ \text{ptr} = \text{new} \ \text{int}[100]; \]

7. Given the function prototype: \( \text{int Cube ( int *)} \); Answer the following question: (Circle the correct answer.)

   Would the function call Cube( &x ) where x is an integer variable in the calling function generate a compile-time error or return some integer
Given the declaration: `int num[5];` Write a for loop to initialize the elements of the array to the values 10, 8, 6, 4, 2 using array-name/offset notation.

```c
for ( int count = 0; count < 5; count++ )
  *( num + count ) = 10 – ( count * 2 );
```

9. Write a C++ statement to change the value of the integer variable pointed to by the pointer variable `myPtr` to 50.

```c
*myPtr = 50;
```

10. Write a C++ statement to declare a variable named `dblPtr` as a pointer to double and assign it the address of double variable `DblVal`.

```c
double * dblPtr = & DblVal;
```

Use the following code segment to answer questions 11 – 15.

```c
1. void Double ( int );
2. int Triple ( int * );
3. int Change ( int * );
4. void main ( )
5. {
6.   int Number = 5;
7.   int Result = 8;
8.   int Var = 2;
9.  Double ( Number );
10.  Result = Triple ( &Number );
11.  Var = Change ( &Number );
12.  Double ( Result );
13.  Result = Triple ( &Var );
14.  }
15. void Double ( int Var )
16. {
17.   Var *= 2;
18. }
19. int Triple ( int * ptr )
20. {
21.   return *ptr * *ptr * *ptr;
22. }
23. int Change ( int *Num )
24. {
25.   static int Z = 5;
26.   Z = Z * 3;
27.   *Num = *Num + Z;
28.   return *Num;
29. }
30. }
31. void main ( )
32. {
33.   int Number = 5;
34.   int Result = 8;
35.   int Var = 2;
36.   Double ( Number );
37.   Result = Triple ( &Number );
38.   Var = Change ( &Number );
39.   Double ( Result );
40.   Result = Triple ( &Var );
41.   }
42. void Double ( int Var )
43. {
44.   Var *= 2;
45. }
46. int Triple ( int * ptr )
47. {
48.   return *ptr * *ptr * *ptr;
49. }
50. int Change ( int *Num )
51. {
52.   static int Z = 5;
53.   Z = Z * 3;
54.   *Num = *Num + Z;
55.   return *Num;
56. }
```

11. What would be the value of the variable `Number` after statement 13 finished executing?

```
5
```

12. What would be the value of the variable `Var` after statement 14 finished executing?

```
20
```

13. What would be the value of the variable `Result` after statement 15 finished executing?

```
125
```

14. What would be the value of the variable `Result` after statement 16 finished executing?

```
20 * 20 * 20 = 8000
```

15. What would be the value of the variable `Z` in function `Change` after statement 16 finished executing?

```
15
```
Programming in C / C++
Working with Arrays, Pointers, and Functions

Use the following information for questions 16 – 19. Read the statements carefully!!!

Given the declarations: \( \text{int } x = 10, \ *xPtr = &x; \)  Assume the address for \( x \) is 1050 and for \( xPtr \) it is 1200. All addresses are decimal rather than hexadecimal for this exercise.

Display the output for the following statements.

16. \( \text{cout } \ll &x; \) 
   \( \text{1050} \)  
17. \( \text{cout } \ll \text{xPtr}; \) 
   \( \text{xPtr } \text{stores the address of } x \) 
18. \( \text{cout } \ll *xPtr * x; \) 
   \( \text{100} \)  \( \text{// }10 * 10\) 
19. \( \text{cout } \ll x; \) 
   \( \text{10} \) 

For questions 20 – 25, circle T (true) or F (false).

20. T F  If the name of an integer array is to be passed to a function, the notation \( \text{int [ ]} \) or \( \text{int *} \) may be used interchangeably to declare the parameter’s type.

21. T F  * is called the indirection operator

22. T F  Array/Subscript and Pointer/Offset notation can be used interchangeably.

23. T F  A pointer must be declared to point to objects of a specific data type.

24. T F  Pointer variables provide another way to pass arguments to a function by reference.

25. T F  For a pointer variable, you can only get information about it’s value or it’s address in memory

For questions 26 – 30, select the best answer.

26. D  To print the value of the pointer variable \( \text{myPtr} \), which statement would you use?
   A. \( \text{cout } \ll &\text{myPtr}; \) 
   B. \( \text{cout } \ll *\text{myPtr}; \) 
   C. \( \text{cout } \ll \text{myPtr} *; \)
   D. \( \text{cout } \ll \text{myPtr}; \)
   E. \( \text{cout } \ll \text{myPtr} &; \)

27. C  Which of the following operations is NOT allowed for a pointer?
   A. adding an integer to a pointer 
   B. dereferencing a pointer 
   C. multiplying a pointer by an integer 
   D. using a pointer variable in a relational expression 
   E. getting the address of a pointer

28. A  Which of the following values can NOT be assigned to a pointer to an integer variable.
   A. The address of a float variable 
   B. 0 (zero) 
   C. NULL 
   D. The address of an integer variable

29. C  Given this function prototype: \( \text{void square ( int *);} \)
   Which of the function calls below would be correct to call function \( \text{square} \) with integer variable \( y \) defined in the calling function.
   A. \( \text{square}[y]; \) 
   B. \( \text{square ( }^*y); \) 
   C. \( \text{square ( }&y); \) 
   D. \( \text{square ( }y); \) 

30. B  Which of the following statements would correctly declare \( \text{Ptr} \) as: a pointer to a double constant?
   A. \( \text{double }\text{ const *Ptr; } \)
   B. \( \text{const }\text{ double }*\text{Ptr; } \)
   C. \( \text{double }\text{Ptr }*\text{ const; } \)
   D. \( \text{double }*\text{ const }\text{Ptr; } \)
   E. \( \text{const }\text{double }*\text{ const }\text{Ptr; } \)