2.4 Assignment 4:

Output:

- Value of a (3)
- 5

No errors with this code

2.5 Assignment 5:

Variable	Last value
*p	50
q	8
*r	8
v	8
*s	50

2.6 Assignment 6:

Variable	**Last Value**
∗**p**	12
∗**q**	11
v	11
nom[**0**]	12
nom[**1**]	13
nom[**2**]	12
nom[**3**]	10
nom[**4**]	16

2.8 Assignment 8:

This code will NOT work and will likely cause a segmentation fault or undefined behavior.

Problems:

- Modifying a string literal's pointer this way leads to unpredictable behavior
- After *x = x[n], x points to '\0', so printf("%s ", x) would print nothing (or crash)
- The x++ continues moving the pointer into undefined memory territory

2.9 Assignment 9:

D. ce

2.10 Assignment 10:

B. 2, 15, 6, 8, 10

Explaination:

- i=0: a[1] = a[0] + 5 = 2 + 5 = 7
- i=1: a[1] = a[1] + 5 = 7 + 5 = 12
- i=2: a[1] = a[2] + 5 = 6 + 5 = 11
- i=3: a[1] = a[3] + 5 = 8 + 5 = 13
- i=4: a[1] = a[4] + 5 = 10 + 5 = 15

2.11 Assignment 11:
B. 20, 4, 4
2.12 Assignment 12:
D. 300
2.13 Assignment 13:
a. *
2.14 Assignment 14:
a. x is a pointer to a string, y is a string.
2.15 Assignment 15:
D. Point to a tye
2.16 Assignment 16:
C. int i; double* dp = &i;
2.17 Assignment 17:
B. p now points to b
2.19 Assignment 19:
A. ABCDEFGHIJ
2.20 Assignment 20:
A. fg
2.23 Assignment 23:
B. const
2.24 Assignment 24:
C. The new operator
2.25 Assignment 25:
B. Indirection
2.26 Assignment 26:
A. sizeof
2.27 Assignment 27:
A. Pointer contains an address of a variable
2.28 Assignmen 28:
C. 3
Explaination: A pointer can be initialized with NULL(0), nullptr, address of a variable
2.29 Assignment 29:
C. Address operator
2.30 Assignment 30:
C. 129, a
2.31 Assignment 31:
D. Compile error
2.32 Assignment 32:
B. 10, 20, 30, 40, 50,
2.33 Assignment 33:
C. 14
2.34 Assignment 34:
C. Compile error
2.35 Assignment 35
Output: 2