2.1. Assignment 1

What is printed out? Are there any problems (errors)?

```
int a = 3;
int *b = &a;

cout << b << endl;
cout << *b << endl;
cout << &b << endl;
cout << a << endl;
cout << &a << endl;
```

Answer:

- The program prints out the address of a, 3 (value stored at the address b points to), the address of b itself, 3 (value of a), and the address of a in five lines respectively.
- There are no errors, but the answer of the addresses of the pointers will be different in each compiler.

2.2. Assignment 2

What is printed out? Are there any problems (errors)?

```
int x,z;
float y;
char ch, *chp;
int *ip1, *ip2;
float *fp;

x = 100;
y = 20.0;
z = 50;
ch = 'Z';

ip1 = &x;
ip2 = &z;
fp = &y;
chp = &ch;

ip2 = ip1;
ip1 = &z;
*ip1 = *ip2;

*ip1 = 200;
*ip1 = *ip2 + 300;
*fp = 1.2;

cout << x << endl;
cout << y << endl;
cout << z << endl;

cout << ip1 << endl;
cout << *ip1 << endl;
cout << &ip1 << endl;

cout << ip2 << endl;
cout << *ip2 << endl;
cout << &ip2 << endl;

cout << fp << endl;
cout << *fp << endl;
cout << &fp << endl;

cout << chp << endl;
cout << *chp << endl;
cout << &chp << endl;
```

Answer:

The program executes as follows:

- `ip1 = &x; ip2 = &z;`
- `ip2 = ip1; ip1 = &z;` (ip2 points to x, ip1 points to z)
- `*ip1 = *ip2;` (z becomes 100 (x = 100))
- `*ip1 = 200;` (z becomes 200)
- `*ip1 = *ip2 + 300;` (z becomes x + 100 = 400)
- `fp = &y; chp = &ch;` (fp points to y, chp points to ch)
- `*fp = 1.2;` (y becomes 1.2)

The printed results:

- The first group (x, y, z): 100, 1.2, 400
- The second group (ip1, *ip1, &ip1): Address of z, 400, Address of ip1
- The third group (ip2, *ip2, &ip2): Address of x, 100, Address of ip2
- The fourth group (fp, *fp, &fp): Address of y, 1.2, Address of fp
- The fifth group (chp, *chp, &chp): Address of ch, 'Z', Address of chp

2.3. Assignment 3

What is printed out? Are there any problems (errors)?

```
int *a = new int;
int *b = new int;
*a = 2;
b = a;
cout << *a << endl;
cout << *b << endl;
delete a;
delete b;
```

Answer:

- The program outputs 2 and 2 (assign a pointing to the address of 2 and b pointing to the same address as a, which is 2).
- The problems in the program:
  + Memory leaking: when assigning b = a, the origial address allocated to b is lost and cannot be deleted.
  + Double free: b and a point to the same address, so deleting the same memory twice may cause a crash.

2.4. Assignment 4

What is printed out? Are there any problems (errors)?

```
int a = 3;
int *p = &a;
cout << *p << endl;
p = new int(5);
cout << *p << endl;
```

Answer:

- The program outputs 3 and 5 (assign p pointing to the address of a (which is 3), and assign p pointing to the new memory containing 5).
- The problems in the program: Memory leaking. The memory allocated in the line `new int(5)` is never released.

2.5. Assignment 5

What are the last values of *p, q, *r, v and *s?

```
int v = 8, *r, *s;
int *p;
int q = 100;
p = &q; (p points to q = 100)
r = p; (r points to q as well)
*p = 20; (q becomes 20)
p = new int; (p points to new memory block)
*r = 30; (q becomes 30)
q = v; (q becomes 8)
s = p; (s points to the memory block p points to)
*s = 50; (The memory block contains 50)
```

Answer:

- The final values are *p = 50, q = 8, *r = 8, v = 8, *s = 50.
- The explanations are contained in the code.

2.6. Assignment 6

What are the last values of *p, *q, v and nom?

```
int *p, *q, v, nom[5];
p = &v; (p points to v)
*p = 12; (v becomes 12)
q = p; (q points to v as well)
nom[0] = *q; (The first value of nom is *q = 12)
p = nom; (p points to the first address of nom (nom[0]))
p++; (p points to the second address of nom (nom[1])
nom[2] = 12;
*p = 13; (nom[1] becomes 13)
*q = 10; (nom[0] becomes 10)
v = 11;
*(p + 3) = 16; (nom[1 + 3] becomes 16)
p = &nom[3]; (p points to nom[3])
*p = 10; (nom[3] becomes 10)
p--; (p points to nom[2])
```

Answer:

- The final values are *p = 12, *q = 11 (points to v), v = 11, nom = {12, 13, 12, 10, 16}.

2.7. Assignment 7

Point out the compile time error in the program given below.

```
#include<stdio.h>

int main()
{
    int *x;
    *x = 100;
    return 0;
}
```

Answer: C. No compile-time error.

2.8. Assignment 8

What will be the output of the program?

```
#include<stdio.h>
#include<string.h>

int main()
{
    int i, n;
    char *x = “Alice”;
    n = strlen(x);
    *x = x[n];
    for(i = 0; i <= n; i++)
    {
        printf(“%s ”, x);
        x++;
    }
    printf(“\n”, x);
    return 0;
}
```

Answer: D. lice ice ce e

Explanation: x initially points to “Alice”, x[n] copies ‘\0’ to the first character. In the loop, the program prints %s after incrementing the pointer x, so it prints ‘\0’ (prints nothing), “lice”, “ice”,…

2.9. Assignment 9

What will be the output of the program?

```
#include<stdio.h>

int main()
{
    char str[] = “peace”;
    char *s = str;
    printf(“%s\n”, s++ +3);
    return 0;
}
```

Answer: D. ce

Explanation: s points to “peace”, s++ is executed after printing, so it added 3 to the address (which is at ‘c’), and prints forwards: “ce”.

2.10. Assignment 10

What will be the output of the program?

```
#include<stdio.h>

int main()
{
    int i, a[] = {2, 4, 6, 8, 10};
    change(a, 5);
    for(i = 0; i <= 4; i++) printf(“%d, ”, a[i]);
    return 0;
}
void change(int *b, int n)
{
    int i;
    for(i = 0; i < n; i++)
        *(b+1) = *(b+i)+5;
}
```

Answer: B. 2, 15, 6, 8, 10

Explanation: The function changes the *(b+1) element, which is b[1], so in the last operation, b[1] is assigned to b[n – 1] + 5 = 15.

2.11. Assignment 11

If the size of the integer if 4 bytes, what will be the output of the program?

```
#include<stdio.h>

int main()
{
    int arr[] = {12, 13, 14, 15, 16};
    printf("%d, %d, %d\n", sizeof(arr), sizeof(*arr),
sizeof(arr[0]));
    return 0;
}
```

Answer: B. 20, 4, 4

Explanation: The size of arr is 4 bytes * 5 = 20 bytes, *arr is 4 (*arr points to the first element of the array), arr[0] is 4 as well.

2.12. Assignment 12

What will be the output of the program?

```
#include<stdio.h>

int main()
{
    char *str;
    str = “%d\n”;
    str++;
    str++;
    printf(str – 2, 300);
    return 0;
}
```

Answer: D. 300

Explanation: str points to “%d\n”, and str++ moves the pointer to ‘d’, str – 2 moves back to ‘%’.

2.13. Assignment 13

The operator used for deferencing or indirection is

A. *

B. &

C. ->

D. ->>

Answer: A. *

2.14. Assignment 14

Choose the right option

```
string* x, y;
```

A. x is a pointer to a string, y is a string
B. y is a pointer to a string, x is a string
C. Both x and y are pointers to string types
D. None of the mentioned

Answer: A.

Explanation: The asterisk associates with x, not the string type.

2.15. Assignment 15

Which one of the following is not a possible state for a pointer?

A. Hold the address of the specific object
B. Point one last the end of an object
C. Zero
D. Point to a type

Answer: D

Explanation: Pointer points to the memory address, not the data type.

2.16. Assignment 16

Which of the following is illegal?

A. `int *p;`

B. `string s, *sp = 0;`

C. `int i; double* dp = &i;`

D. `int *pi = 0;`

Answer: C

Explanation: The dp pointer is assigned to point to a double variable, so as it points to i, it is illegal.

2.17. Assignment 17

What will happen in this code?

```
int a = 100, b = 200;
int *p = &a, *q = &b;
p = q;
```

A. b is assigned to a
B. p now points to b
C. a is assigned to b
D. q now points to a

Answer: B

2.18. Assignment 18

What is the output of this program?

```
#include <iostream>
using namespace std;
int main()
{
    int a = 5, b = 10, c = 15;
    int *arr[] = {&a, &b, &c};
    cout << arr[1];
    return 0;
}
```

A. 5
B. 10
C. 15
D. It will return some random number

Answer: D

Explanation: arr[1] is &b, which is the address of b, so it prints the memory address.

2.19. Assignment 19

What is the output of this program?

```
#include<iostream>
using namespace std;
int main()
{
    char arr[20];
    int i;
    for (i = 0; i < 10; i++)
        *(arr + i) = 65 + i;
    *(arr + i) = ‘\0’;
    cout << arr;
    return 0;
}
```

A. ABCDEFGHIJ
B. AAAAAAAAAA
C. JJJJJJJJJ
D. None of the mentioned

Answer: A

Explanation: 65 is the ASCII code of ‘A’, so the loop fills the array with ‘A’ through ‘J’. The next line place the terminating null after ‘J’.

2.20. Assignment 20

What is the output of this program?

```
#include<iostream>
using namespace std;
int main()
{
   char *ptr;
   char Str[] = {abcdefg};
   ptr = Str;
   ptr += 5;
   cout << ptr;
   return 0;
}
```

A. fg
B. cdef
C. defg
D. abcd

Answer: A

Explanation: ptr points to the first element of Str, which is ‘a’. (ptr + 5) moves to the index 5 (‘f’), and it prints from ‘f’ to end.

2.21. Assignment 21

A pointer can be initialized with

A. Null

B. Zero

C. Address of an object of same type

D. All of them

Answer: D

2.22. Assignment 22

Which from following is not a correct way to pass a pointer to a function?

A. Non-constant pointer to non-constant data
B. A non-constant pointer to constant data
C. A constant pointer to non-constant data
D. None of them

Answer: D

2.23. Assignment 23

A qualifer that enables programmers to inform compiler that value of a particular variable should not be modified?

A. ptr
B. const
C. stsrt
D. None of them

Answer: B

2.24. Assignment 24

Which operator return address of unallocated blocks in memory?

A. The delete operator
B. The empty operator
C. The new operator
D. All of them

Answer: C

2.25. Assignment 25

Referencing a value through a pointer is called

A. Direct calling
B. Indirection
C. Pointer referencing
D. All of them

Answer: B

2.26. Assignment 26

Which unary operator is used for determining the size of an array?

A. sizeof
B. size_array
C. s_array
D. size_ofarray

Answer: A

2.27. Assignment 27

What is a pointer?

A. Pointer contains an address of a variable
B. It’s an operator
C. It’s a function
D. None of them

Answer: A

2.28. Assignment 28

How many values can be used to initialize a pointer?

A. 1
B. 2
C. 3
D. 4

Answer: C (0, nullptr, address)

2.29. Assignment 29

A unary operator that returns an address of its operands, are called

A. Pointer operator
B. Relationship operator
C. Address operator
D. Both A and B

Answer: C (&)

2.30. Assignment 30

What will be the output of the following program?

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

int main()
{
    int a = 32, *ptr = &a;
    char ch = ‘A’, &cho = ch;

    cho += a;
    *ptr += ch;
    cout << a << “, ” << ch << endl;
    return 0;
}
```

A. 32, A
B. 32, a
C. 129, a
D. 129, A

Answer: C

Explanation:

- cho += a means cho = 97, so ch becomes 97 (‘a’).
- *ptr += ch means a + ch = 129.