5. Program set 4

1. Pointer example

```
#include <iostream>
int main()
{
    int x = 42;
    int* p = &x;                 // &x = address of x

    std::cout << "x=" << x << std::endl;
    std::cout << "p (addr)=" << p << std::endl;
    std::cout << "*p (value)=" << *p << std::endl;  // dereference
    *p = 99;                                    // write through pointer
    std::cout << "x after *p=99 -> " << x << std::endl;

    return 0;
}
```

2. Pass by pointer

```
#include <iostream>
void squareByPtr(int* numPtr)
{
    *numPtr = (*numPtr) * (*numPtr);
}

int main()
{
    int a=5;
    squareByPtr(&a);
    std::cout << a << std::endl;

    return 0;
}
```

3. const varieties of pointers

```
#include <iostream>
int main()
{
    int a = 10, b = 20;

    const int* p1 = &a;     // pointer to const int (data read-only via p1)
    // *p1 = 5;             // ERROR if uncommented
    p1 = &b;                // pointer itself can change

    int* const p2 = &a;     // const pointer to int (address fixed)
    *p2 = 11;               // data modifiable
    // p2 = &b;             // ERROR if uncommented

    const int* const p3 = &a;  // const pointer to const int
```

```cpp
    std::cout << *p1 << " " << *p2 << " " << *p3 << std::endl;

    return 0;
}
```

4. null pointer

```cpp
#include <iostream>
int main()
{
    int* p = 0;                         // null pointer
    if (p == 0)
        std::cout << "p is null" << std::endl;

    int x = 7;
    p = &x;

    if (p)
        std::cout << "*p=" << *p << std::endl;

    return 0;
}
```

5. dangling pointer

```cpp
#include <iostream>

int* badFunc()
{
    int local = 4;
    return &local;
}  // returns address of a dead local!

int main()
{
    int* q = badFunc();        // q is dangling; do NOT dereference
    std::cout << "Received a dangling pointer (demo only)." << std::endl;

    return 0;
}
```

6. arrays and pointers

```cpp
#include <iostream>
int main()
{
    int a[4] = {10,20,30,40};
    int* p = a;                       // points to first element
    std::cout << a[0] << " " << *p << std::endl;

    return 0;
}
```

7. pointer arithmetic

```cpp
int main()
{
    long arr[4] = {6,0,9,6};
    long* ptr = arr;
    std::cout << "*ptr=" << *ptr << std::endl;   // 6

    ++ptr;                                 // moves by sizeof(long)
    std::cout << "*ptr=" << *ptr << std::endl;   // 0

    std::cout << arr[2] << " " << *(arr+2) << std::endl;
    std::cout << arr[2] << " " << *(ptr+1) << std::endl;

    return 0;
}
```

8. pointer subtraction

```cpp
#include <iostream>
int main()
{
    long arr[4] = {6,0,9,6};

    long* p = arr;
    long* q = arr + 3;

    std::cout << "q - p = " << (q - p) << std::endl;  // 3 elements apart

    return 0;
}
```

9. pointer subscript

```cpp
#include <iostream>
int main()
{
    int a[5] = {2,4,6,8,10};
    int* p = a;

    std::cout << p[3] << " " << *(p+3) << " " << a[3] << std::endl; // 8 8 8

    return 0;
}
```

10. walking through an array using pointer

```cpp
#include <iostream>
int main()
{
    int a[5] = {1,2,3,4,5};

    for (int* p = a; p != a + 5; ++p)
        std::cout << *p << " ";

    std::cout << std::endl;

    return 0;
}
```

11. walking in reverse through an array using pointer

```
#include <iostream>
int main()
{
    int a[5] = {1,2,3,4,5};

    for (int* p = a + 5; p-- != a; )
        std::cout << *p << " ";

    std::cout << std::endl;

    return 0;
}
```

12. string array vs string literal

```
#include <iostream>

int main()
{
    char course1[] = { '6','.', '0','9','6','\0' }; // modifiable array
    const char* course2 = "6.096";                  // string literal (read-only)

    course1[1] = 'X';                               // OK
    // course2[1] = 'X';                            // RUNTIME ERROR if attempted
    std::cout << course1 << " | " << course2 << std::endl;

    return 0;
}
```

13. length of string

```
#include <iostream>
#include <cstddef>

std::size_t my_strlen(const char* s)
{
    const char* p = s;

    while (*p)
        ++p;
    return std::size_t(p - s);   // distance in elements
}

int main()
{
    const char* s="hello";
    std::cout << my_strlen(s) << std::endl;

    return 0;
}
```

14. string copy

```
#include <iostream>

void my_strcpy(char* dst, const char* src)
{
    while ((*dst++ = *src++))
    {
        /* copy including '\0' */
    }
}

int main()
{
    char buf[32];

    my_strcpy(buf, "Pointers!");

    std::cout << buf << std::endl;

    return 0;
}
```

15. find character in string

```
#include <iostream>

const char* my_strchr(const char* s, int ch)
{
    for (; *s; ++s)
        if (*s == ch)
            return s;

    return 0;
}

int main()
{
    const char* s="hello";
    const char* p=my_strchr(s,'l');

    std::cout << (p ? p : "(not found)") << std::endl;

    return 0;
}
```

16. pass array and array size to function

```
#include <iostream>

int sum(const int* a, int n)
{
    int s=0;

    for(int i=0;i<n;++i)
        s+=a[i];

    return s;
}

int main()
{
    int a[5]={1,2,3,4,5};

    std::cout << sum(a,5) << std::endl;

    return 0;
}
```

17. using n[a] and a[n]

```
#include <iostream>

int main()
{
    int a[5] = {10,20,30,40,50};

    std::cout << a[3] << " == " << 3[a] << std::endl;  // both 40

    return 0;
}
```