1. How do you initialize a pointer?

2. What is the difference between ptr++ and *ptr++?

3. How do you pass pointers to a function?

4. What is a null pointer?

5. How do you allocate memory dynamically using pointers?

6. Explain pointer arithmetic.

7. What are function pointers?

8. What happens if you dereference a wild pointer?

9. What is a dangling pointer?

10. How can you create a pointer to a pointer?

11. What is the difference between int* ptr; and int *ptr;?

12. What is a void pointer?

13. How do you swap two integers using pointers?

14. Explain why we need to use the free() function.

15. What is the purpose of the sizeof operator when used with pointers?

16. What will happen if you attempt to access memory that has been freed?

17. How can you create an array of pointers?

18. What is the difference between an array and a pointer?

19. How do you return a pointer from a function?

20. Can you explain how multi-dimensional arrays work with pointers?

21. Write a C program to reverse an array using pointers.

22. Write a C program to dynamically allocate memory for an array of integers.

23. Write a C function using void * pointers to create a generic swap function that can swap two variables of any type.

24. Write a C program using function pointers to switch between addition and multiplication functions.

Predict the output of the following questions:

1.
```
#include <stdio.h>
int main() {
    int x = 10;
    int *p = &x;
    printf("%d\n", *p);
    *p = 20;
    printf("%d\n", x);
    return 0;
}
```

2.
```
#include <stdio.h>
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int *p = arr;
    printf("%d\n", *(p + 2));
    printf("%d\n", *p++);
```

```
    printf("%d\n", *p);
    return 0;
}
```

3.
```
#include <stdio.h>

int main() {
    int *p = NULL;
    if (p) {
        printf("Pointer is not null\n");
    } else {
        printf("Pointer is null\n");
    }
    return 0;
}
```

4.
```
#include <stdio.h>

int main() {
    int x = 100;
    int *p = &x;
    int **q = &p;
    printf("%d\n", **q);
    return 0;
}
```

5
```
#include <stdio.h>

int main() {
```

```
    int arr[] = {10, 20, 30, 40};
    int *p = arr;
    printf("%d\n", *p + 1);
    printf("%d\n", *(p + 1));
    printf("%d\n", *(arr + 2));
    return 0;
}
```

6.
```
#include <stdio.h>

int main() {
    int a = 10;
    int b = 20;
    int *p = &a;
    printf("%d\n", *p);
    p = &b;
    printf("%d\n", *p);
    return 0;
}
```

7.
```
#include <stdio.h>

int main() {
    int x = 10, y = 20;
    int *const p = &x;
    printf("%d\n", *p);
    *p = y;
    printf("%d\n", *p);
    return 0;
```

```
}
```

8.
```
#include <stdio.h>

int main() {
   char str[] = "Hello";
   char *p = str;
   printf("%c\n", *p);
   printf("%c\n", *(p + 1));
   printf("%s\n", p + 2);
   return 0;
}
```

9.
```
#include <stdio.h>

int main() {
   int a = 10, b = 20, c = 30;
   int *arr[] = {&a, &b, &c};
   printf("%d\n", *arr[0]);
   printf("%d\n", *arr[1]);
   printf("%d\n", *arr[2]);
   return 0;
}
```

10.
```
#include <stdio.h>

void display(int x) {
   printf("%d\n", x);
}

int main() {
```

```
    void (*funcPtr)(int) = display;
    funcPtr(5);
    return 0;
}
```

11.
```
#include <stdio.h>

int main() {
    int x = 10;
    int *p = &x;
    printf("%d\n", *p);
    (*p)++;
    printf("%d\n", *p);
    return 0;
}
```

12.
```
#include <stdio.h>

int main() {
    int arr[] = {5, 10, 15};
    int *p = arr;
    for (int i = 0; i < 3; i++) {
        printf("%d ", *(p + i));
    }
    printf("\n");
    return 0;
}
```