Data Structure

Name: Mohammed Al-zubairi

1. What is the difference between Type* pointer_name; and Type *pointer_name?

DataType* pointer_name; and DataType *pointer_name; are syntactically equivalent. The difference between them lies purely in the style and readability preference of the programmer.

Explanation:

int* pointer_name: This style suggests that the type of the variable ***pointer_name*** is a ***pointer to Type***. Some programmers prefer this style because it visually groups the `*` with the type, indicating that the variable is of a pointer type.

DataType *pointer_name: In this style, the `*` is placed next to the variable name, which can be interpreted as the variable ***pointer_name*** is a pointer to `Type`". This is the style commonly seen in many C and C++ codebases, as it aligns with the way multiple pointers can be declared in the same line, like ***Type *pointer1, *pointer2.***

Key Point:

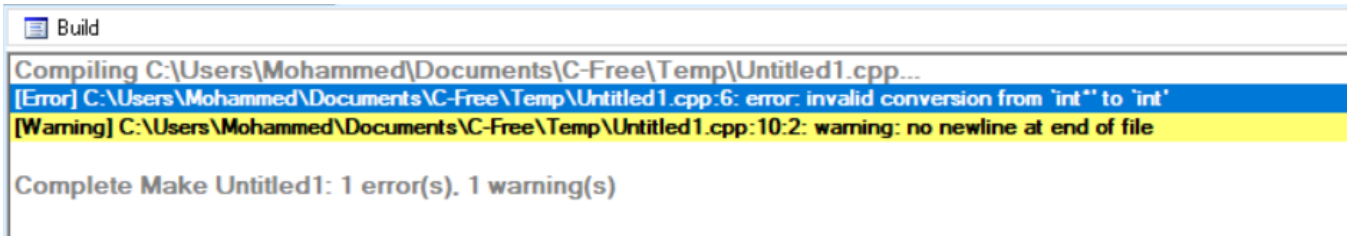
The placement of the `*` does not change the meaning of the code. Both styles will compile to the same machine code, and the choice of which style to use is up to the individual or team coding standards.

2. **Make a screenshot for the error that may occur if you assigned a value of type pointer (&a) to the p variable.**

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

int main(){
	int a = 100;
	int p = &a;

	cout << a << "  " << &a << endl;
	cout << p << "  " << &p << endl;
}
```



The error that occurs is that the compiler was unable to convert from int pointer to int.

3. Write a different program with the same variables that make the same output.

The program:

```
int main(){
	int a = 100, b = 88, c = 8;
	int *p1 = &a, *p2, *p3 = &c;
	p2 = &b;
	p2 = p1;
	b = *p3;
	*p2 = *p3;
	cout << a << b << c << endl;

	//THE OUTBUT IS ( 888 )
}
```

The result:

```
int main(){
        int a = 100, b = 88, c = 8;
        int *p1 = &a, *p2, *p3 = &c;

        p1 = &c;
        p2 = p3;
        p3 = &b;



        cout <<  *p1 << *p3 << endl;

        //THE OUTBUT IS ( 888 )
}
```

4. Write a different program with the same variables that make the same output.

The program:

```
void doubleIt (int x, int *p){
        *p = 2 * x;
}

int main(){

        int a = 16;
        doubleIt(9, &a);

        return 0;
}
```

The second program:

```
void doubleIt (int x, int *p){
        *p = x + (*p - x) + 2;
        cout << *p << endl;
}
```

```
int main(){

	int a = 16;
	doubleIt(9, &a);

	return 0;
}
```

5. Write a different program with the same variables that make the same output.

Delete the (*) from the parameter of the method and the (&) from the argument of the object in the main of this code:

```
void doubleIt (int x, int *p){
	*p = x + (*p - x) + 2;
	cout << *p << endl;
}

int main(){

	int a = 16;
	doubleIt(9, &a);

	return 0;
}
```

The result: