GDB Debugger:

Practice 1:

First directly run the program and see whether it can run normally. And the GDB debugger shows: 5 *ptr = 10;   // Segmentation fault occurs here. Because the ptr is a nullptr, we are not allowed to visit a memory space of a nullptr, so we get a segmentation fault.

```
Fixed Code: int main() {
        int* ptr = new int;    /        *ptr = 10;
        std::cout << "Value: " << *ptr << std::endl;
        delete ptr;
        return 0;
```

Practice 2:

We use static analysis to find this error.

```
Fixed Code: #include <iostream>
int main() {
        int* arr = new int[100];
        // Properly delete the allocated memory
        delete[] arr;
        return 0;
}
```

Practice 3:

First directly run the code, find the program will sort the variable in array from big to small. The biggest one will be the left of the array, and the smallest one will be the right of the array. And then see the code, find the name of the function is bubble sort, bubble sort should make the biggest one to the right of the array, so this function is wrong.

Fixed Code:

```
# void bubbleSort(int arr[], int n) {
        for (int i = 0; i < n - 1; i++) {
                for (int j = 0; j < n - i - 1; j++) {
                        if (arr[j] > arr[j + 1]) {
                                std::swap(arr[j], arr[j + 1]);
                                }
                        }
                }
        }
```

Practice 4:

1. Run the program and find infinite loop,
2. Set breakpoints, print the value of n, and find that when the value of n is less than 0, and there is no stop condition.
3. Add a stop condition, if n&lt;0, stop running
4. Run the program again and it can run

Fixed Code: #include <iostream>

```
// Recursive function without a base case
int recursiveFunction(int n){
        std::cout <<"n="<<n<< std::endl;
        if(n <= 0){
                return 0; //or any other appropriate return value}
        return recursiveFunction(n-1);}//Infinite recursion
int main(){
        recursiveFunction(5);
        return 0;
        }
```