Grupa B:

1.
2. słowo kluczowa mutable + podać przykład kodu
3. klasa abstrakcyjna
4. STL - uzupełnij kod
5. Kod:

```
// uzupełnij

struct A {
    // uzupełnij
};

int main() {
    list<int> coll(11);
    for_each( /*uzupełnij*/ );
    copy( /*uzupełnij*/ );
    return 0;
}

//to ma się wyświetlić:
// -1; -2; -3; -4; -5; -6; -7; -8; -9; -10; -11;
```

21.
22. czy poniższy kod jest poprawny? jeśli tak, to co się wyświetli + uzasadnienie
23. Kod:

```
#include <iostream>

using namespace std;

struct A { ~A() { cout << "~A\n"; } };
struct B { ~B() { cout << "~B\n"; } };
struct X : virtual public A, private B { ~X() { cout << "~X\n"; } };
struct Y : virtual public A, private B { ~Y() { cout << "~Y\n"; } };
struct Z : public X, public Y { ~Z() { cout << "~Z\n"; } };


int main() {
    Z test;
    return 0;
}
```

39.

kod jest poprawny, wyświetli się:

40. Kod:

```
~Z ~Y ~B ~X ~B ~A
```

42. (oczywiście tam są nowe linie, ale szkoda miejsca, uzasadnienie sobie można wymyślić samemu, warto wspomnieć o virtualnym dziedziczeniu)

_____

Panie Iskra, tam było:

Kod:

```
struct Z : public Y, public X { Z() { cout << "Z\n"; } };
```

co w oczywisty sposób zmienia kolejność konstrukcji da tej hierarchii.

z grupy B:

Kod:

```
#include <iostream>
#include <algorithm>
#include <list>
#include <iterator>

using namespace std;

int liczba = -1;

struct A {
    A() { }

    void operator()(int &element)
    {
        element = liczba;
        liczba--;
    }
};

int main() {
    list<int> coll(11);
```

```
    for_each(coll.begin(), coll.end(), A());
    copy(coll.begin(), coll.end(), ostream_iterator<int>(cout, "; "));
    return 0;
}
```