Grupa B:

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

   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;
```

4. czy poniższy kod jest poprawny? jeśli tak, to co się wyświetli + uzasadnienie

   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;
}
```

   kod jest poprawny, wyświetli się:

   Kod:

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

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

Grupa A:
1. Co to jest typedef + przykłady
2. Dziedziczenie- co to jest - rodzaje -kiedy warto dziedziczyć, a kiedy nie?
3. STL - uzupełnij kod:
Kod:

```
// uzupełnij

struct A {
   // uzupełnij
};

int main() {
   list<deque> coll(13);
   for_each( /*uzupełnij*/ );
   copy( /*uzupełnij*/ );
   return 0;
}

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

4.czy poniższy kod jest poprawny? jeśli tak, to co się wyświetli + uzasadnienie
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 Y, public X { Z() { cout << "Z\n"; } };


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