1. Schemat blokowy:
* czy można zapisać program łamiący zasady programowania strukturalnego (TAK)
* nie można zapisać rekurencji (NIE)
* Można zapisać dowolny program / można zapisać wszystkie algorytmy (TAK)

2. Wzorzec projektowy vs klasa
* 100% kodu można sprawdzić (NIE)
* większa korzyść ze wzorca (TAK)
* Łatwiej zastosować wzorzec niż klasę (TAK)

3. Refaktoryzacja
* optymalizuje kod (NIE)
* reinżynieria (NIE)
* przebudowa kodu (TAK)

4. Na którym poziomie CMM istnieje już 6 praktyk
* 3 poziom rozwoju (TAK)
* 4 poziom rozwoju (NIE)
* 5 poziom rozwoju (NIE)

5. Testowanie vs debukowanie
* ten sam cel (NIE)
* różny cel (TAK)
* mogą być stosowane zamiennie (NIE)
„testowanie ma na celu wykrycie błędów, debugowanie ma na celu lokalizację i poprawę błędów”

6. XP vs Crystal Cases
* XP to większa swoboda (NIE)
* XP wymaga większej dyscypliny (wiekse ograniczenia) (TAK)
* Crystal Cases łatwiejszy do zastosowania (TAK)

7. Programowanie strukturalne, a proceduralne
*strukturalne jest rozwinięciem proceduralnego ( )
* strukturalne jest poddyscypliną proceduralnego (TAK)
* ??? (NIE)

8. Obiekt:
* jest unikalny (TAK)
* charakteryzuje się zachowaniem (TAK)
* Charakteryzuje się stanem (TAK)

9. Spaghetti code
* określenie pejoratywne (TAK)
* łatwe w utrzymaniu i trudne do zrozumienia (NIE)
* polega na zapisywaniu nazw zmiennych po włosku (NIE)

10. Testowanie metodą białej skrzynki
* wymaga znajomości algorytmu (TAK)
* teoretycznie w 100% kodu można sprawdzić (TAK)
* jest tańsze od czarnej skrzynki (TAK)

11. Ograniczenia w projektowaniu
* są dobrą praktyką (TAK)
* upraszczają kod (TAK)
* są niepotrzebne (NIE)

12. Dobrze użyty przypadek użycia
* łatwy do zrozumienia (TAK)
* opisuje wymagania niefunkcjonalne (NIE)
* definiuje format danych (NIE)

13. Throwaway project
• Zaczynamy od tego czego nie rozumiemy (Tak)
• Zaczynamy od tego co rozumiemy (Nie)
• Może być produktem ostatecznym (Nie)

14. Stuprocentowe pokrycie kodu
• Zawsze możliwe (Nie)
• Nie zawsze możliwe (Tak)
• Daje gwarancję najwyższej jakości (Nie)

15. 4+1 View
• Występuje tylko w UML (Nie)
• Wszystkie perspektywy (Tak)
• Perspektywa tylko logiczna (Nie)

16. Programowanie obiektowe
• Hierarchia + modularność (Tak)
• Abstrakcja + enkapsulacja (Tak)
• Polimorfizm + dziedziczenie (Tak)

17. XP
• Programowanie parami (Tak)
• Większa swoboda ( )
• Integralność (Tak)

18. Iteracyjność
• Tylko w Agile (Nie)
• Mało błędów (Tak)
• Umożliwia wczesne wdrożenie (Tak)
• Przez częste wdrożenie, będzie działać (Tak)

19. Metody formalne
• Nie wymagają dokumentacji (Nie)
• Są drogie (Tak)
• Nie wymagane testowanie (Nie)

20. Analiza dynamiczna
• Funkcjonalne
• Niefunkcjonalne
• Żadne z wyżej wymienionych

21. Analiza strukturalna
• modele danych (TAK)
• funkcji (TAK)
• zachowania systemu (TAK)

22. Przypadek użycia
* łatwy do zrozumienia (TAK)
* opisuje wymagania funkcjonalne (TAK)
* definiuje format danych (NIE)

23. Diagram encja-obiekt
* definiuje związek między obiektami danych
*
*

24. Diagram przepływu danych
* pisuje sposób przetwarzania danych
*przedstawia procedury przetwarzające dane