Nazwa kwalifikacji:	**Projektowanie, programowanie i testowanie aplikacji**
Oznaczenie kwalifikacji:	**INF.04**
Numer zadania:	**01**
Kod arkusza:	**INF.04-01-24.06-SG**
Wersja arkusza:	**SG**

Lp.	Elementy podlegające ocenie/kryteria oceny
R.1	**Rezultat 1: Implementacja, kompilacja, uruchomienie programu**
	Uwaga: kryteria należy odnieść do aplikacji konsolowej, jeżeli ta nie istnieje, zastosować 1.1 ÷ 1.6 do aplikacji mobilnej. Kryteria dotyczą wyłącznie samodzielnie napisanego kodu. Wystarczy, że sprawdzaną cechę zastosowano dla większości przypadków w kodzie
R.1.1	Kod źródłowy zapisano w sposób czytelny: instrukcje w osobnych liniach, stosowane spacje pomiędzy operatorami, konsekwentnie stosowana wybrana konwencja dla nawiasów klamrowych
R.1.2	Kod zapisano z wcięciami dla zagnieżdżeń bloków
R.1.3	Użyto polskie lub angielskie, znaczące nazewnictwo funkcji (metod)
R.1.4	Użyto polskie lub angielskie, znaczące nazewnictwo zmiennych (pól). Wyjątkami od reguły są zmienne: bufor, tmp, iteratory pętli itp. Kryterium **nie jest** spełnione tylko wtedy, gdy nazwy zmiennych nic nie znaczą, np.: x, fun, foo, tab, tablica, …
R.1.5	Zastosowano typy zmiennych pasujące do problemu (np. tablica dowolnego typu całkowitego, zmienne całkowite dla punktów, liczby losowań, zmienna znakowa dla zgody 't'/'n') W przypadku języka Python, tam gdzie jest to wymagane, zastosowano jawną konwersję do odpowiednich typów
R.1.6	Podjęto próbę skompilowania lub interpretowania kodu, co udokumentowano zrzutem ekranowym przedstawiającym uruchomiony program ew. kompilację lub skrypt
R.1.7	Program nawiązuje zrozumiałą komunikację z użytkownikiem: monit o wprowadzenie danych, wyprowadzanie wyników opatrzone komentarzem. Jeżeli kod nie uruchamia się z powodu błędów kompilacji - sprawdzić w kodzie aplikacji
R.2	**Rezultat 2: Aplikacja konsolowa**
	Uwaga: kryteria 2.1 ÷ 2.6 należy sprawdzić w kodzie programu, sprawdzane elementy muszą być zapisane zgodnie ze składnią. *Gdy aplikacja nie uruchamia się, a zdający zapisał zrzuty ekranu z uruchomienia aplikacji należy sprawdzić powód braku kompilacji. Jeśli występują błędy w plikach źródłowych zdającego, kryteria 2.7 ÷ 2.10 nie są spełnione. Jeżeli błędy występują w innych plikach należy ocenić na podstawie kodu i zrzutu ekranu*
R.2.1	Program składa się z programu głównego oraz przynajmniej dwóch funkcji (metod): losującej i liczącej punkty (mogą być niedokończone, lub z błędami) oraz zastosowano zmienną typu tablicowego dla liczb całkowitych, wypełnioną wartościami całkowitymi od 1 do 6
R.2.2	Funkcja/metoda licząca punkty jest typu liczbowego całkowitego i zawiera zapisaną instrukcję zwracającą wartość
R.2.3	Wyświetlanie zawartości tablicy i wypełnianie jej wartościami losowymi zostało zaimplementowane w oparciu o pętlę. Pętla ma właściwą liczbę iteracji
R.2.4	Zastosowano przynajmniej jedną pętlę do wprowadzania liczby kostek lub sprawdzania zgody na dalsze działanie programu
R.2.5	Liczby są losowane z przedziału od 1 do 6. Losowane jest tyle liczb, ile kostek wskazał użytkownik. Losowanie odbywa się w metodzie / funkcji. W podejściu strukturalnym funkcja musi mieć parametry wejściowe (nie są stosowane zmienne globalne)
R.2.6	Punkty są liczone jako suma oczek, dla tych wartości, które zostały wylosowane przynajmniej dwa razy
R.2.7	Program kompiluje się i uruchamia w konsoli, co udokumentowano zrzutem ekranu. Pytanie o liczbę kostek jest powtarzane tak długo, aż zostanie podana liczba z zakresu 3..10 (sprawdzić dla granicznych: 2 i 11 oraz 3 i 10)

R.2.8	Program wyświetla tyle liczb z zakresu 1..6 ile użytkownik podał na początku działania aplikacji
R.2.9	Program wyświetla obliczoną sumę punktów
R.2.10	Podanie 't' kontynuuje grę, wybranie 'n' przerywa grę
R.3	**Rezultat 3: Aplikacja mobilna**
	Uwaga: należy uwzględnić różnice pomiędzy emulacjami - nie należy brać pod uwagę takich cech jak marginesy, wielkości bloków, itp. Kryteria 3.1 ÷ 3.6 sprawdzić w kodzie źródłowym, sprawdzane elementy muszą być zapisane zgodnie ze składnią. *Gdy aplikacja nie uruchamia się, a zdający zapisał zrzuty ekranu z uruchomienia aplikacji należy sprawdzić powód braku kompilacji. Jeśli występują błędy w plikach źródłowych zdającego kryteria 3.7 ÷ 3.10 nie są spełnione. Jeżeli błędy występują w innych plikach lub bibliotekach sprawdzić w kodzie oraz na zrzucie ekranu. Dla Android Studio dopuszcza się także rozwiązanie w języku Kotlin*
R.3.1	Zastosowano język znaczników XML/XAML lub inny do opisu interfejsu użytkownika oraz zastosowano rozkład liniowy wertykalny (LinearLayout lub StackLayout, lub VerticalStackLayout, lub inny o tej idei) z zagłębionym rozkładem liniowym horyzontalnym dla obrazów
R.3.2	Umieszczono napis "Gra w kości. Autor" z numerem zdającego, dwa przyciski, pięć obrazów wypełnionych grafiką *question.jpg* oraz dwa napisy
R.3.3	Zastosowano kolory tła: rozkład wertykalny lub strona: Beige (#F5F5DC), napis z tytułem: Brown (#A52A2A), oba przyciski: Chocolate (#D2691E), rozkład horyzontalny: biały oraz biały kolor czcionki dla napisu z tytułem gry (w MAUI nazwy kolorów obowiązkowo wielką literą)
R.3.4	Dobrano marginesy, odstępy i wielkość grafiki tak aby kontrolki nie stykały się ze sobą oraz było widoczne w całości 5 obrazów, napis z tytułem oraz przyciski są wyśrodkowane, pozostałe wypełniają w poziomie całą szerokość strony (np. fill, fillAndExpand, match_parent)
R.3.5	Program obsługuje przynajmniej jedno zdarzenie kliknięcia przycisku w sposób właściwy dla danego środowiska programistycznego. Wystarczy, że kontrolka przycisku jest odpowiednio powiązana ze zdarzeniem
R.3.6	Program odwołuje się do kontrolek w sposób właściwy dla danego środowiska programistycznego. Wystarczy, że zastosowano poprawnie dla jednej kontrolki
R.3.7	Po wciśnięciu przycisku RZUĆ KOŚĆMI wyświetlane są wylosowane liczby w postaci 5 obrazów kości, każdy obraz odpowiada wyrzuconej liczbie oczek (w uruchomionej aplikacji lub na zrzucie i obowiązkowo w kodzie)
R.3.8	Po wciśnięciu przycisku RZUĆ KOŚĆMI wyświetlana jest liczba punktów za wykonany rzut (kryterium poprawne również, gdy algorytm nieprawidłowo zaimplementowany) oraz wyświetlany jest wynik gry, który jest powiększony o wynik aktualnego rzutu (w uruchomionej aplikacji lub na zrzucie i obowiązkowo w kodzie)
R.3.9	Po wciśnięciu przycisku „RESETUJ WYNIK” zerowany jest wynik gry oraz wyświetlane są oba wyniki jako 0 (w uruchomionej aplikacji lub na zrzucie i obowiązkowo w kodzie)
R.3.10	Aplikacja kompiluje się i uruchamia w emulatorze, co udokumentowano zrzutem ekranu. Jej układ jest zgodny z obrazem 3 w arkuszu egzaminacyjnym
R.4	**Rezultat 4: Dokumentacja aplikacji**
	Uwaga: nagłówek z kryteriów 4.1 ÷ 4.5 musi być zgodny ze stanem faktycznym z kodu źródłowego, nawet jeżeli w kodzie są błędy logiczne (liczba pól, typy). *Zrzuty ekranu z kryteriów 4.6 i 4.7 muszą zawierać cały obszar ekranu z widocznym paskiem zadań.* *Dokumentacja z kryterium 4.8 zapisana jest w pliku egzamin*
R.4.1	Dla jednej z funkcji/metody z aplikacji konsolowej zapisano nagłówek w postaci komentarza zgodny z Listingiem 1 z arkusza egzaminacyjnego (nie liczymy gwiazdek), komentarz może być wieloliniowy lub kilka jednoliniowych lub Docstrings (potrójny cudzysłów) - w tym przypadku opis znajduje się pod nazwą funkcji
R.4.2	W komentarzu ujęto nazwę i opis działania funkcji / metody
R.4.3	W komentarzu ujęto nazwy i opis wszystkich parametrów funkcji / metody

R.4.4	W przypadku gdy funkcja / metoda zwraca wartość, ujęto w komentarzu opis tej wartości. W przeciwnym przypadku zapisano void, brak itp.
R.4.5	W komentarzu ujęto numer zdającego
R.4.6	Zapisano przynajmniej jeden zrzut ekranu z uruchomienia lub kompilacji aplikacji konsolowej, na zrzucie widoczne jest środowisko, w którym powstała aplikacja
R.4.7	Zapisano przynajmniej jeden zrzut ekranu z uruchomienia lub kompilacji aplikacji mobilnej, na zrzucie widoczne jest środowisko, w którym powstała aplikacja
R.4.8	Dokumentacja zawiera nazwy: systemu operacyjnego, środowisk programistycznych, emulatora oraz języków programowania