Załącznik nr 1.5 do Zarządzenia Rektora UR nr 12/2019

SYLABUS

DOTYCZY CYKLU KSZTAŁCENIA *2020-2024*
(skrajne daty)
Rok akademicki 2020/2021 i 2021/2022

1. PODSTAWOWE INFORMACJE O PRZEDMIOCIE

Nazwa przedmiotu	***Algorytmy i struktury danych***
Kod przedmiotu*	
Nazwa jednostki prowadzącej kierunek	*Kolegium Nauk Przyrodniczych*
Nazwa jednostki realizującej przedmiot	*Kolegium Nauk Przyrodniczych, Instytut Informatyki*
Kierunek studiów	*Informatyka i ekonometria*
Poziom studiów	*studia I stopnia*
Profil	*praktyczny*
Forma studiów	*stacjonarne*
Rok i semestr/y studiów	*rok I, II, semestr 2, 3*
Rodzaj przedmiotu	*przedmiot kierunkowy*
Język wykładowy	*język polski*
Koordynator	*dr hab. Jan G. Bazan, prof. UR*
Imię i nazwisko osoby prowadzącej / osób prowadzących	

***** *-opcjonalnie, zgodnie z ustaleniami w Jednostce*

1.1.Formy zajęć dydaktycznych, wymiar godzin i punktów ECTS

Semestr (nr)	Wykł.	Ćw.	Konw.	Lab.	Sem.	ZP	Prakt.	Inne (jakie?)	**Liczba pkt. ECTS**
2	30	15							3
3	30			15					3

1.2. Sposób realizacji zajęć

☒zajęcia w formie tradycyjnej

☐ zajęcia realizowane z wykorzystaniem metod i technik kształcenia na odległość

1.3 Forma zaliczenia przedmiotu (z toku) (egzamin, zaliczenie z oceną, zaliczenie bez oceny)

ZALICZENIE Z OCENĘ PO SEM. 2 I EGZAMIN PO SEM. 3

2.WYMAGANIA WSTĘPNE

Elementy logiki i teorii mnogości, Analiza matematyczna, Algebra liniowa z geometrią analityczną, Podstawy programowania

3. CELE, EFEKTY UCZENIA SIĘ , TREŚCI PROGRAMOWE I STOSOWANE METODY DYDAKTYCZNE

3.1 Cele przedmiotu

C1	Nauczenie studentów metod konstrukcji algorytmów, metod analizy ich złożoności obliczeniowej oraz metod analizy ich poprawności.
C2	Nauczenie studentów podstawowych struktur danych, metod ich implementacji i wykorzystania w praktyce.
C3	Zapoznanie studentów z przykładową biblioteką standardowych struktur danych.

3.2 Efekty uczenia się dla przedmiotu

EK (efekt uczenia się)	Treść efektu uczenia się - **wiedza minimalna i minimalne umiejętności** do zaliczenia	Odniesienie do efektów kierunkowych
EK_01	Student zna notacje asymptotyczne, metody wykorzystywania ich do wyznaczania złożoności obliczeniowej algorytmów oraz techniki obliczeniowe pozwalające poprawnie wyznaczać złożoność obliczeniową (czasową i pamięciową) dla algorytmów iteracyjnych. Jednak nie zna w wystarczającym stopniu technik obliczeniowych pozwalających na wyznaczanie złożoności dla algorytmów rekurencyjnych. Zna podstawowe klasy złożoności obliczeniowej algorytmów, ale nie zawsze potrafi je porównać z punktu widzenia złożoności obliczeniowej oraz poprawnie ocenić ich praktyczne znaczenie do rozwiązywania rzeczywistych problemów algorytmicznych	K_W01, K_W02
EK _02	Student zna abstrakcyjne struktury danych, metody ich implementacji w przynajmniej jednym języku programowania oraz gotowe implementacje w dedykowanej bibliotece standardowej, w tym stosy, kolejki, listy, drzewa, grafy, słowniki, haszowanie, drzewa przeszukiwań binarnych. W szczególności student zna budowę tych struktur oraz operacje jakie mogą być wykonywane na tych strukturach. Jednakże posiadana wiedza o efektywności tych operacji w kontekście złożoności obliczeniowej nie pozwala mu na w pełni poprawne porównanie tych struktur pod względem ich efektywności obliczeniowej oraz na dobieranie struktur danych do ustalonych wymagań dotyczących efektywności obliczeniowej algorytmów.	K_W01
EK_03	Student zna zasady formułowania i algorytmizacji zadań oraz notację zapisu algorytmów w pseudojęzyku i w wybranym języku programowania, a także podstawowe techniki i metody projektowania i implementowania algorytmów, w tym metodę dynamicznego przydziału pamięci, rekurencję, metodę brutalnej siły, metodę dziel i zwyciężaj, programowanie dynamiczne, algorytmy zachłanne, metodę Monte Carlo, przeszukiwanie z nawrotami. Zna także podstawowe algorytmy wyszukiwania i sortowania. Jednak nie zawsze potrafi poprawnie porównać te metody i algorytmy pod względem efektywności czasowej i dokładności otrzymanego rozwiązania. Nie zawsze potrafi także dobrze dobierać metody i algorytmy	K_W01

	do wymagań oczekiwanej efektywności rozwiązania danego problemu	
EK_04	Student zna metodę dowodzenia semantycznej poprawności algorytmów, w tym poszczególne jej kroki (dowód częściowej poprawności, dowód własności określoności i dowód własności stopu) oraz rozumie te kroki. Nie zna jednak wszystkich metod dowodzenia tych kroków oraz nie zna uzasadnienia praktycznego dla zasadności każdego z tych kroków.	K_W01, K_W02
EK_05	Student umie poprawnie śledzić algorytm bez rekurencji zapisany w wybranym języku programowania lub w tzw. pseudojęzyku.	K_U01, K_U02
EK_06	Student potrafi zastosować abstrakcyjne typy danych do rozwiązywania problemów z użyciem języka programowania, przy czym zawsze poprawnie operuje na strukturach danych przechowujących tylko wartości typów konkretnych	K_U01
EK_07	Student potrafi poprawnie wyznaczać złożoność obliczeniową algorytmów (czasową i pamięciową) przy wykorzystaniu notacji asymptotycznych dla algorytmów iteracyjnych. Jednak nie zawsze potrafi poprawnie wyznaczać złożoności obliczeniowej algorytmów dla algorytmów rekurencyjnych	K_U01, K_U02, K_K02
EK_08	Student potrafi poprawnie wykorzystać podstawowe techniki i metody projektowania i implementowania algorytmów, w tym metodę dynamicznego przydziału pamięci, rekurencję, metodę brutalnej siły, metodę dziel i zwyciężaj, programowanie dynamiczne, algorytmy zachłanne, metodę Monte Carlo, przeszukiwanie z nawrotami oraz algorytmy sortowania, wyszukiwania i przeszukiwania grafów. Zaprojektowane i zaimplementowane przez studenta algorytmy zawsze posiadają własność stopu. Jednak algorytm skonstruowany przez studenta podczas weryfikacji efektu kształcenia ma dwa błędy umożliwiające uznanie go za poprawny z punktu widzenia określoności operacji stosowanych w algorytmie lub poprawności uzyskanych na wyjściu algorytmu wyników.	K_U01, K_U02
EK_09	Student weryfikuje semantyczną poprawność algorytmów w takim zakresie, że potrafi poprawnie zdefiniować warunki alfa i beta częściowej poprawności algorytmu oraz poprawnie formułuje warunek stopu i warunek określoności algorytmu. Nie zawsze jednak potrafi poprawnie udowodnić częściową poprawność oraz warunki stopu i określoności algorytmu	K_U01, K_U02, K_K02

3.3 Treści programowe

A. Problematyka wykładu

Treści merytoryczne
Część pierwsza
1. Ogólne wprowadzenie do przedmiotu. Pseudojęzyk. Śledzenie algorytmów bez rekurencji.
2. Złożoność obliczeniowa algorytmów bez rekurencji.
3. Wybrane metody rozwiązywanie równań rekurencyjnych.
4. Algorytmy z rekurencją i ich śledzenie.
5. Złożoność obliczeniowa algorytmów z rekurencją.
6. Semantyczna poprawność algorytmów.

7. Praktyczny aspekt zagadnienia poprawności semantycznej w programowaniu (asercje, testy jednostkowe, dzienniki)
Część druga
8. Wprowadzenie do metod konstruowania algorytmów.
9. Algorytmy przeszukiwania wyczerpującego
10. Metoda dziel i zwyciężaj oraz programowanie dynamiczne
11. Algorytmy aproksymacyjne.
12. Abstrakcyjne struktury danych (lista, zbiór, drzewo, graf, słownik).
13. Konkretne struktury danych (tablica dynamiczna, lista powiązana, drzewo binarne, tablica mieszająca) .
14. Metody implementacji abstrakcyjnych struktur danych.
15. Podstawowe algorytmy wyszukiwania i sortowania.
16. Implementacja grafów i wybrane algorytmy grafowe.
17. Trudność problemów.

B. Problematyka ćwiczeń audytoryjnych, konwersatoryjnych, laboratoryjnych, zajęć praktycznych

Treści merytoryczne
Część pierwsza
1. Zadania na konstruowanie i śledzenie algorytmów bez rekurencji.
2. Zadania na wyznaczanie złożoności obliczeniowej algorytmów bez rekurencji.
3. Zadania na rozwiązywanie równań rekurencyjnych.
4. Zadania na konstruowanie i śledzenie algorytmów rekurencyjnych.
5. Zadania na wyznaczanie złożoności obliczeniowej algorytmów rekurencyjnych.
6. Zadania na dowodzenie poprawności semantycznej algorytmów.
Część druga
7. Zadanie na konstrukcję algorytmów metodami przeszukiwania wyczerpującego.
8. Zadanie na konstrukcję algorytmów metodami przeszukiwania z nawrotami.
9. Zadanie na konstrukcję algorytmów metodami dziel i zwyciężaj oraz na programowanie dynamiczne.
10. Zadania na konstrukcję algorytmów zachłannych
11. Zadania na konstrukcję algorytmów metodami stochastycznymi.
12. Zadania na implementację abstrakcyjnych struktur danych, także z użyciem standardowych struktur danych.
13. Zadania na wyszukiwanie i sortowanie, także z użyciem standardowych implementacji algorytmów dostępnych w bibliotece standardowych struktur danych.
14. Przykłady problemów należących do klas P, NP, NP-trudne, NP-zupełne.

3.4 Metody dydaktyczne

Wykład: wykład z prezentacją multimedialną
Ćwiczenia: rozwiązywanie zadań "tablicowych"
Laboratorium: rozwiązywanie zadań, w tym programistycznych z użyciem komputera.

4. METODY I KRYTERIA OCENY

4.1 Sposoby weryfikacji efektów uczenia się

Symbol efektu	Metody oceny efektów uczenia się (np.: kolokwium, egzamin ustny, egzamin pisemny, projekt, sprawozdanie, obserwacja w trakcie zajęć)	Forma zajęć dydaktycznych (w, ćw, ...)
EK_ 01	Kolokwium pisemne, egzamin pisemny	W, ĆW

EK_ 02	Kolokwium pisemne, egzamin pisemny	W, ĆW
EK_ 03	Kolokwium pisemne, egzamin pisemny	W, LAB
EK_ 04	Kolokwium pisemne, egzamin pisemny	W, ĆW
EK_ 05	Kolokwium pisemne	W, ĆW
EK_ 06	Kolokwium przy komputerze	W, LAB
EK_ 07	Kolokwium pisemne	W, ĆW
EK_ 08	Kolokwium przy komputerze	W, LAB
EK_ 09	Kolokwium pisemne	W, ĆW

4.2 Warunki zaliczenia przedmiotu (kryteria oceniania)

Efekt	Ocena	Kryteria otrzymania oceny
EK_01	dst	Student zna notacje asymptotyczne, metody wykorzystywania ich do wyznaczania złożoności obliczeniowej algorytmów oraz techniki obliczeniowe pozwalające poprawnie wyznaczać złożoność obliczeniową (czasową i pamięciową) tylko dla algorytmów iteracyjnych. Nie zna lub zna w niewystarczającym stopniu techniki obliczeniowe pozwalające na wyznaczanie złożoności dla algorytmów rekurencyjnych. Zna podstawowe klasy złożoności obliczeniowej algorytmów, ale nie zawsze potrafi je porównać z punktu widzenia złożoności obliczeniowej oraz poprawnie ocenić ich praktyczne znaczenie do rozwiązywania rzeczywistych problemów algorytmicznych.
	db	Student zna notacje asymptotyczne, metody wykorzystywania ich do wyznaczania złożoności obliczeniowej algorytmów, ale zna niezbędne techniki obliczeniowe pozwalające poprawnie wyznaczać złożoność obliczeniową (czasową i pamięciową) tylko dla algorytmów iteracyjnych oraz dla algorytmów z rekurencją prostą. Nie zna lub zna w niewystarczającym stopniu techniki obliczeniowe pozwalające na wyznaczanie złożoności dla algorytmów z rekurencją rozgałęzioną. Zna podstawowe klasy złożoności obliczeniowej algorytmów oraz potrafi je porównać z punktu widzenia złożoności obliczeniowej, ale nie zawsze potrafi poprawnie ocenić ich praktyczne znaczenie do rozwiązywania rzeczywistych problemów algorytmicznych.
	bdb	Student zna notacje asymptotyczne, metody wykorzystywania ich do wyznaczania złożoności obliczeniowej algorytmów, w tym niezbędne techniki obliczeniowe pozwalające poprawnie wyznaczać złożoność obliczeniową (czasową i pamięciową) zarówno dla algorytmów iteracyjnych, jak i dla algorytmów z rekurencją (w tym prostą i rozgałęzioną). Zna podstawowe klasy złożoności obliczeniowej algorytmów, potrafi je porównać z punktu widzenia

		złożoności obliczeniowej oraz potrafi ocenić ich praktyczne znaczenie do rozwiązywania rzeczywistych problemów algorytmicznych.
EK_02	dst	Student zna abstrakcyjne struktury danych, metody ich implementacji w przynajmniej jednym języku programowania oraz gotowe implementacje w dedykowanej bibliotece standardowej, w tym stosy, kolejki, listy, drzewa, grafy, słowniki, haszowanie, drzewa przeszukiwań binarnych. W szczególności student zna budowę tych struktur oraz operacje jakie mogą być wykonywane na tych strukturach. Jednakże posiadana wiedza o efektywności tych operacji w konkretnych implementacjach tych struktur w kontekście złożoności obliczeniowej, nie pozwala mu na w pełni poprawne porównanie tych struktur pod względem ich efektywności obliczeniowej oraz na dobieranie struktur danych do ustalonych wymagań dotyczących efektywności obliczeniowej algorytmów.
	db	Student zna abstrakcyjne struktury danych, metody ich implementacji w przynajmniej jednym języku programowania oraz gotowe implementacje w dedykowanej bibliotece standardowej, w tym stosy, kolejki, listy, drzewa, grafy, słowniki, haszowanie, drzewa przeszukiwań binarnych. W szczególności student zna budowę tych struktur oraz operacje jakie mogą być wykonywane na tych strukturach. Ponadto, ma wiedzę o efektywności tych operacji w kontekście złożoności obliczeniowej, co daje mu możliwość porównania tych struktur pod względem ich efektywności obliczeniowej. Nie potrafi jednak w pełni poprawnie dobierać struktur danych do ustalonych wymagań dotyczących efektywności obliczeniowej rozwiązania danego problemu algorytmicznego.
	bdb	Student zna abstrakcyjne struktury danych, metody ich implementacji w przynajmniej jednym języku programowania oraz gotowe implementacje w dedykowanej bibliotece standardowej, w tym stosy, kolejki, listy, drzewa, grafy, słowniki, haszowanie, drzewa przeszukiwań binarnych. W szczególności student zna budowę tych struktur oraz operacje jakie mogą być wykonywane na tych strukturach. Ponadto, ma wiedzę o efektywności tych operacji w kontekście złożoności obliczeniowej, co daje mu możliwość porównania tych struktur pod względem ich efektywności obliczeniowej oraz dobierania struktur danych do ustalonych wymagań dotyczących efektywności obliczeniowej.
EK_03	dst	Student zna zasady formułowania i algorytmizacji zadań oraz notację zapisu algorytmów w pseudojęzyku i w wybranym języku programowania, a także podstawowe techniki i metody projektowania i implementowania algorytmów, w tym metodę dynamicznego przydziału pamięci, rekurencję, metodę brutalnej siły, metodę dziel i zwyciężaj, programowanie dynamiczne, algorytmy zachłanne, metodę Monte Carlo, przeszukiwanie z nawrotami. Zna także podstawowe algorytmy wyszukiwania i sortowania. Jednak nie zawsze potrafi poprawnie porównać te metody algorytmy pod względem efektywności czasowej i dokładności otrzymanego rozwiązania. Nie zawsze

		potrafi także dobrze dobierać metody i algorytmy do wymagań oczekiwanej efektywności rozwiązania danego problemu.	
	db	Student zna zasady formułowania i algorytmizacji zadań oraz notację zapisu algorytmów w pseudojęzyku i w wybranym języku programowania, a także podstawowe techniki i metody projektowania i implementowania algorytmów, w tym metodę dynamicznego przydziału pamięci, rekurencję, metodę brutalnej siły, metodę dziel i zwyciężaj, programowanie dynamiczne, algorytmy zachłanne, metodę Monte Carlo, przeszukiwanie z nawrotami. Zna także podstawowe algorytmy wyszukiwania i sortowania. Potrafi porównać te metody i algorytmy pod względem efektywności czasowej i dokładności otrzymanego rozwiązania. Nie zawsze potrafi jednak dobrze dobierać metody i algorytmy do wymagań oczekiwanej efektywności rozwiązania danego problemu.	
	bdb	Student zna zasady formułowania i algorytmizacji zadań oraz notację zapisu algorytmów w pseudojęzyku i w wybranym języku programowania, a także podstawowe techniki i metody projektowania i implementowania algorytmów, w tym metodę dynamicznego przydziału pamięci, rekurencję, metodę brutalnej siły, metodę dziel i zwyciężaj, programowanie dynamiczne, algorytmy zachłanne, metodę Monte Carlo, przeszukiwanie z nawrotami. Zna także podstawowe algorytmy wyszukiwania i sortowania. Potrafi porównać te metody i algorytmy pod względem efektywności czasowej i dokładności otrzymanego rozwiązania, a także poprawnie dobiera metody do wymagań oczekiwanej efektywności rozwiązania danego problemu.	
EK_04	dst	Student umie śledzić algorytm iteracyjny bez rekurencji zapisany w wybranym języku programowania lub w tzw. pseudojęzyku.	
	db	Student umie śledzić algorytm iteracyjny bez rekurencji rozgałęzionej zapisany w wybranym języku programowania lub w tzw. pseudojęzyku.	
	bdb	Student umie śledzić algorytm iteracyjny oraz rekurencyjny (w tym z rekurencją rozgałęzioną) zapisany w wybranym języku programowania lub w tzw. pseudojęzyku.	
EK_05	dst	Student potrafi zastosować abstrakcyjne typy danych do rozwiązywania problemów z użyciem języka programowania, przy czym operuje na strukturach danych przechowujących tylko wartości typów prostych (np. int, byte, float, double, char, boolean itd.)	
	db	Student potrafi zastosować abstrakcyjne typy danych do rozwiązywania problemów z użyciem języka programowania, przy czym operuje na strukturach danych przechowujących tylko wartości typów obiektowych-zdefiniowanych (np. String, Integer, Double, itd.).	
	bdb	Student potrafi zastosować abstrakcyjne typy danych do rozwiązywania problemów z użyciem języka programowania, przy czym operuje na strukturach danych przechowujących wartości dowolnych typów	

		obiektowych, w tym typów zdefiniowanych przez studenta (np. Osoba, Punkt, itd.).
EK_06	dst	Student potrafi poprawnie wyznaczać złożoność obliczeniową algorytmów (czasową i pamięciową) przy wykorzystaniu notacji asymptotycznych dla algorytmów iteracyjnych. Nie potrafi poprawnie wyznaczać złożoności obliczeniowej algorytmów dla algorytmów rekurencyjnych.
	db	Student potrafi poprawnie wyznaczać złożoność obliczeniową algorytmów (czasową i pamięciową) przy wykorzystaniu notacji asymptotycznych dla algorytmów iteracyjnych oraz dla algorytmów rekurencyjnych bez rekurencji rozgałęzionej. Nie potrafi poprawnie wyznaczać złożoności obliczeniowej algorytmów dla algorytmów z rekurencją rozgałęzioną.
	bdb	Student potrafi poprawnie wyznaczać złożoność obliczeniową algorytmów (czasową i pamięciową) przy wykorzystaniu notacji asymptotycznych dla algorytmów iteracyjnych oraz dla algorytmów rekurencyjnych (w tym z rekurencją rozgałęzioną).
EK_07	dst	Student potrafi poprawnie wykorzystać podstawowe techniki i metody projektowania i implementowania algorytmów, w tym metodę dynamicznego przydziału pamięci, rekurencję, metodę brutalnej siły, metodę dziel i zwyciężaj, programowanie dynamiczne, algorytmy zachłanne, metodę Monte Carlo, przeszukiwanie z nawrotami oraz algorytmy sortowania, wyszukiwania i przeszukiwania grafów. Zaprojektowane i zaimplementowane przez studenta algorytmy zawsze posiadają własność stopu. Jednak algorytm skonstruowany przez studenta podczas weryfikacji efektu uczenia się ma 2 błędy umożliwiające uznanie go za poprawny z punktu widzenia określoności operacji stosowanych w algorytmie lub poprawności uzyskanych na wyjściu algorytmu wyników.
	db	Student potrafi poprawnie wykorzystać podstawowe techniki i metody projektowania i implementowania algorytmów, w tym metodę dynamicznego przydziału pamięci, rekurencję, metodę brutalnej siły, metodę dziel i zwyciężaj, programowanie dynamiczne, algorytmy zachłanne, metodę Monte Carlo, przeszukiwanie z nawrotami oraz algorytmy sortowania, wyszukiwania i przeszukiwania grafów. Zaprojektowane i zaimplementowane przez studenta algorytmy zawsze posiadają własność stopu. Jednak algorytm skonstruowany przez studenta podczas weryfikacji efektu uczenia się ma jeden błąd umożliwiający uznanie go za poprawny z punktu widzenia określoności operacji stosowanych w algorytmie lub poprawności uzyskanych na wyjściu algorytmu wyników.
	bdb	Student potrafi poprawnie wykorzystać podstawowe techniki i metody projektowania i implementowania algorytmów, w tym metodę dynamicznego przydziału pamięci, rekurencję, metodę brutalnej siły, metodę dziel i zwyciężaj, programowanie dynamiczne, algorytmy zachłanne, metodę Monte Carlo, przeszukiwanie z nawrotami oraz algorytmy sortowania,

		wyszukiwania i przeszukiwania grafów. Zaprojektowane i zaimplementowane przez studenta algorytmy zawsze posiadają własność stopu. Ponadto, algorytm skonstruowany przez studenta podczas weryfikacji efektu uczenia się jest poprawny z punktu widzenia określoności operacji stosowanych w algorytmie oraz poprawności uzyskanych na wyjściu algorytmu wyników.

Zasady uzyskania oceny końcowej:

Zaliczenie ćwiczeń i laboratorium następuje na podstawie zaliczenia wszystkich efektów weryfikowanych przez planowane w danym okresie metody weryfikacji. Przy czym zakłada się, że każda metoda weryfikacji dostarcza osobne oceny dla każdego z weryfikowanych przez nią efektów uczenia się. Jeśli dany efekt jest weryfikowany przez więcej niż jedną metodę, to ocena weryfikująca osiągnięcie tego efektu jest obliczana jako średnia arytmetyczna ocen uzyskanych w poszczególnych metodach weryfikowania tego efektu.

Student otrzymuje z zaliczenia ocenę **niedostateczny,** gdy metody weryfikacji wykażą, iż co najmniej jeden z efektów nie został osiągnięty (średnia ocena dla tego efektu jest niższa niż 3.0);

Student otrzymuje ocenę **dostateczny,** gdy przeciętnie każdy z efektów zostanie osiągnięty na poziomie co najmniej 3.0, ale chociaż jeden z efektów został osiągnięty na poziomie mniejszym od 3.75;

Student otrzymuje ocenę **dobry,** gdy przeciętnie każdy z efektów zostanie osiągnięty na poziomie co najmniej 3.75, ale chociaż jeden z efektów został osiągnięty na poziomie mniejszym od 4.75;

Student otrzymuje ocenę **bardzo dobry,** gdy przeciętnie każdy z efektów zostanie osiągnięty na poziomie co najmniej 4.75;

Zaliczenie przedmiotu następuje na podstawie oceny uzyskanej na egzaminie

Student otrzymuje ocenę **niedostateczny,** gdy nie zaliczył ćwiczeń lub egzamin wykaże, iż co najmniej jeden z efektów nie został osiągnięty (średnia ocena dla tego efektu jest niższa niż 3.0);

Student otrzymuje ocenę **dostateczny,** gdy posiada zaliczenie z ćwiczeń, a przeciętnie każdy z efektów weryfikowanych na egzaminie zostanie osiągnięty na poziomie co najmniej 3.0, ale chociaż jeden z efektów został osiągnięty na poziomie mniejszym od 3.75;

Student otrzymuje ocenę **dobry,** gdy posiada zaliczenie z ćwiczeń oraz przeciętna ocena z zaliczenia każdego efektu spośród weryfikowanych na egzaminie wyniesie co najmniej 3.75, ale chociaż jeden z efektów został osiągnięty na poziomie mniejszym od 4.75;

Student otrzymuje ocenę **bardzo dobry,** gdy posiada zaliczenie z ćwiczeń oraz przeciętna ocena z zaliczenia każdego efektu spośród weryfikowanych na egzaminie wyniesie co najmniej 4.75.

5. CAŁKOWITY NAKŁAD PRACY STUDENTA POTRZEBNY DO OSIĄGNIĘCIA ZAŁOŻONYCH EFEKTÓW W GODZINACH ORAZ PUNKTACH ECTS

Forma aktywności	Średnia liczba godzin na zrealizowanie aktywności
Godziny kontaktowe wynikające z harmonogramu studiów	90
Inne z udziałem nauczyciela akademickiego (udział w konsultacjach, egzaminie)	2
Godziny niekontaktowe – praca własna studenta (przygotowanie do zajęć, egzaminu, napisanie referatu itp.)	58
SUMA GODZIN	150
SUMARYCZNA LICZBA PUNKTÓW ECTS	6

** Należy uwzględnić, że 1 pkt ECTS odpowiada 25-30 godzin całkowitego nakładu pracy studenta.*

6. PRAKTYKI ZAWODOWE W RAMACH PRZEDMIOTU

wymiar godzinowy	-
zasady i formy odbywania praktyk	-

7. LITERATURA

LITERATURA PODSTAWOWA:

1. Aho, A., V., Hopcroft, J.,E., Ullman, J., D.: Algorytmy i struktury danych, Helion (2003).
2. Banachowski, L., Diks, K., Rytter, W.: *Algorytmy i struktury danych*, WNT (2006).
3. Cormen, T., H., Leiserson, C. E, Rivest, R., L., Stein, C.: *Wprowadzenie do algorytmów*, WNT (2003 lub nowsze).
4. Lafore, R.: *Java: algorytmy i struktury danych*, Helion (2003).

LITERATURA UZUPEŁNIAJĄCA:

1. Wróblewski , P: *Algorytmy: struktury danych i techniki programowania*, Helion (2003 lub nowsze).
2. LIPSKI W., *KOMBINATORYKA DLA PROGRAMISTÓW*, WNT (DOWOLNE WYDANIE)

Akceptacja Kierownika Jednostki lub osoby upoważnionej