Plan wynikowy z wymaganiami edukacyjnymi

Temat (rozumiany jako lekcja)	Wymagania konieczne (ocena dopuszczająca)	Wymagania podstawowe (ocena dostateczna)	Wymagania rozszerzające (ocena dobra)	Wymagania dopełniające (ocena bardzo dobra)	Wymagania wykraczające (ocena celująca)
I Algorytmy					
Definicja algorytmu (1)	Uczeń: - podaje jedną z definicji algorytmu - umie określić wynik działania bardzo prostego algorytmu podanego przez nauczyciela - wie czym zajmuje się algorytmika	Uczeń: - zna co najmniej dwie definicje algorytmu i potrafi powiedzieć czym się one różnią - podaje przykład prostego algorytmu i omawia jego działanie	Uczeń: - rozumie pojęcie algorytmu "skończonego" - wymienia podstawowe elementy algorytmu - wymienia cechy elementów algorytmu - podaje przykład algorytmu opisującego zjawisko z jego otoczenia	Uczeń: - umie uszczegółowić definicję algorytmu - samodzielnie podaje przykład algorytmu rozwiązania zagadnienia matematycznego (lub z innej dziedziny) i omawia jego działanie - rozwija poznane definicje w oparciu o literaturę lub publikacje	
Dane wejściowe i wyjściowe - oraz związek między nimi (2)	Uczeń: - wymienia rodzaje danych wyjściowych i wejściowych - z pomocą nauczyciela podaje najważniejsze cechy poszczególnych rodzajów danych - rozróżnia różne rodzaje danych wejściowych i wyjściowych oraz definiuje je z pomocą nauczyciela	Uczeń: - samodzielnie definiuje dane wejściowe i wyjściowe różnych typów - określa rodzaje danych wejściowych i wyjściowych dla prostych przykładów algorytmów - omawia rolę algorytmu w powstawaniu danych wyjściowych	Uczeń: - samodzielnie określa rodzaje danych wejściowych i wyjściowy na podstawie analizy działania prostego algorytmu - podaje proste przykłady danych wejściowych i wyjściowych w powiązaniu z nieskomplikowanymi algorytmami	Uczeń: - samodzielnie określa rodzaje i zakres danych wejściowych i wyjściowy na podstawie analizy algorytmu opisującego zjawisko związane matematyczne lub fizyczne - przewiduje rodzaj i zakres (rodzaju) danych wyjściowych na podstawie logarytmu i zakresu (rodzaju) danych wejściowych	Uczeń: - podaje przykłady złożonych algorytmów opisujących zjawiska z otoczenia lub zagadnienia z różnych dziedzin nauki np. fizyki i matematyki określając dokładnie rodzaj i zakres danych wejściowych i wyjściowych
Specyfikacja	Uczeń:	Uczeń:	Uczeń:	Uczeń:	Uczeń:

algorytmu (3)	- podaje definicję specyfikacji algorytmu - z pomocą nauczyciela umie określić funkcję i znaczenie specyfikacji algorytmu - wie jakie zagadnienia musi określić specyfikacja algorytmu	- podaje cechy dobrego zapisu specyfikacji algorytmu - wymienia najważniejsze składowe specyfikacji algorytmu - podaje przykład specyfikacji prostego algorytmu	- opisuje znaczenie specyfikacji algorytmu w procesie powstawania programu komputerowego - poprawnie interpretuje gotowe specyfikacje algorytmu - wskazuje istotne elementy specyfikacji algorytmu	- opisuje znaczenie specyfikacji algorytmu w procesie powstawania algorytmu nad którym pracuje zespół - określa znaczenie poszczególnych elementów specyfikacji algorytmu - sporządza przykładową specyfikację algorytmu	- samodzielnie sporządza specyfikację danego algorytmu
Sposoby zapisywania algorytmów - lista kroków i pseudokod (4)	Uczeń: - definiuje pojęcie "lista kroków" - definiuje pojęcie "pseudokod"	Uczeń: - wie czym jest lista kroków i potrafi wskazać zalety i wady takiego zapisu - rozumie pojęcie pseudokodu i podaje wady i zalety takiego zapisu algorytmu	Uczeń: - umie opisać algorytmem w postaci listy kroków opisane przez nauczyciela zjawisko lub zagadnienie matematyczne - umie opisać z pomocą nauczyciela, algorytmem w postaci pseudokodu opisane przez nauczyciela zjawisko lub zagadnienie matematyczne - wymienia istotne różnice pomiędzy pseudokodem a listą kroków	Uczeń: - umie samodzielnie ułożyć algorytm i zapisać go w postaci listy kroków - umie samodzielnie ułożyć algorytm i zapisać go w postaci pseudokodu	Uczeń: - samodzielnie układa algorytmy w obu postaciach, zawierające wiele warunków
Sposoby zapisywania algorytmów - schemat blokowy (5)	Uczeń: - umie narysować i nazwać wszystkie bloki schematu blokowego - umie określić ilość wejść i wyjść poszczególnych bloków i ją uzasadnić - umie wskazać poprawne i niedozwolone rodzaje połączeń pomiędzy	Uczeń: - opisuje podstawowe przeznaczenie poszczególnych bloków w tym bloku decyzyjnego - wie jaka jest różnica pomiędzy blokiem wejścia/wyjścia, a blokiem operacyjnym - zna zasady łączenia bloków w schemacie	Uczeń: - umie zamienić algorytm z postaci listy kroków lub pseudokodu na postać schematu blokowego - układa prosty algorytm w postaci schematu blokowego poprawnie stosując wszystkie zasady - używa programu komputerowego do układania schematów blokowych	Uczeń: - samodzielnie układa algorytmy opisujące podane przez nauczyciela zjawiska lub problemy matematyczne, fizyczne itp. - testuje i poprawia błędy w schematach blokowych - biegle posługuje się edytorem schematów blokowych	Uczeń: - samodzielnie układa algorytmy w postaci schematów blokowych zawierające kilka bloków decyzyjnych

	blokami algorytmu	blokowym	- opisuje na przykładach znaczenie bloku decyzyjnego		
Złożoność obliczeniowa - podstawy (6) Złożoność obliczeniowa - zasada dużego O (7)	Uczeń: - wie czym jest złożoność obliczeniowa i jakie ma znaczenie dla oceny jakości algorytmu - wymienia wartości składające się na złożoność obliczeniową związane z pamięcią i czasem - umie wymienić nazwy złożoności dużego O	Uczeń: - definiuje złożoność pamięciową i czasową - wie od czego zależy złożoność czasowa - wie od czego zależy złożoność pamięciowa - wie o czym świadczy rząd funkcji wyrażającej złożoność	Uczeń: - umie wstępnie oszacować złożoność pamięciową algorytmu - umie wstępnie oszacować złożoność czasową algorytmu - określa wpływ złożoności czasowej i obliczeniowej na szybkość wykonywania algorytmu - umie określić od czego zależy złożoność algorytmu i wskazuje istotne dla niej "miejsca" algorytmu - umie określić zmienia się liczba operacji dominujących w trakcie działania programu/algorytmu - rozumie pojęcia opisujące notację dużego O dotyczące złożoności obliczeniowej	Uczeń: - określa wpływ złożoności czasowej i obliczeniowej na szybkość wykonywania algorytmu - umie określić stopień złożoności czasowej i określa go jako pesymistyczną, oczekiwaną lub optymistyczną - umie określić zmienia się liczba operacji dominujących w trakcie działania programu/algorytmu - umie powiązać zachowanie się liczby operacji z poszczególnymi rzędami złożoności	Uczeń: - umie określać złożoność obliczeniową na etapie tworzenia algorytmu i wprowadza korekty mające na celu jej poprawę - umie zdefiniować złożoność sześcienną, wielomianową i wykładniczą
Algorytmy na liczbach całkowitych					
Jak zbadać, czy dana liczba jest liczbą pierwszą? (8)	Uczeń: - zna pojęcie liczby pierwszej - umie wymienić kilkanaście liczb pierwszych - wie jakie znaczenie ma pierwiastek kwadratowy dla	Uczeń: - umie, na podstawie gotowego algorytmu (lista kroków), zbadać czy dana liczba jest liczbą pierwszą - rozumie metodę badania użytą w algorytmie	Uczeń: - umie, na podstawie gotowego algorytmu (schemat blokowy), zbadać czy dana liczba jest liczbą pierwszą - omawia metodę zastosowaną w przykładowym algorytmie z	Uczeń: - rozumie metody generowania liczb pierwszych przedstawione przez Legendre`a, Eulera lub Escotta - modyfikuje algorytm i program (P_7_1.pas) dodając elementy generujące komunikaty i	Uczeń: - samodzielnie układa algorytmy i programy generujące liczby pierwsze wg metod Legendre`a, Eulera lub Escotta

	zbadania czy liczba jest liczbą pierwszą		podręcznika - rozumie działanie programu ułożonego na podstawie algorytmu (P_7_1.pas) - testuje działanie programu	wyświetlające wyniki. - dokonuje optymalizacji programu (P_7_1.pas) wykluczając na początku funkcji liczby parzyste	
Jak zbadać czy dana liczba jest liczbą doskonałą? (9)	Uczeń: - zna pojęcie liczby doskonałej - umie wymienić kilkanaście liczb doskonałych	Uczeń: - umie wymienić 2 liczby doskonałe i dowieść że nimi są - umie, na podstawie gotowego algorytmu (lista kroków), zbadać czy dana liczba jest liczbą doskonałą - rozumie metodę badania użytą w algorytmie	Uczeń: - umie, na podstawie gotowego algorytmu zbadać czy dana liczba jest liczbą doskonałą - omawia metodę zastosowaną w przykładowym algorytmie z podręcznika - rozumie działanie programu ułożonego na podstawie algorytmu (P_8_1.pas) - testuje działanie programu	Uczeń: - modyfikuje algorytm i program (P_8_1.pas) dodając elementy wyświetlające także podzielniki danej liczby oraz ich sumę. - układa algorytm badania czy liczba jest doskonała według alternatywnej metody greckich matematyków	Uczeń: - układa program na podstawie alternatywnego algorytmu badania czy liczba jest doskonała
Jak rozłożyć liczbę na czynniki pierwsze? (10)	Uczeń: - wie czym są czynniki złożone i pierwsze - zna definicję rozkładu liczby na czynniki pierwsze	Uczeń: - rozumie metodę rozkładu na czynniki pierwsze - korzystając z gotowego algorytmu umie rozłożyć liczbę na czynniki pierwsze	Uczeń: - umie samodzielnie ułożyć algorytm rozkładu liczby na czynniki pierwsze i zapisać go w postaci listy i schematu blokowego	Uczeń: - umie ułożyć program na podstawie opracowanego przez siebie algorytmu realizujący rozkład liczby na czynniki pierwsze - modyfikuje algorytm i program a taki sposób by zabezpieczyć go przed wprowadzaniem liczb <=1	Uczeń: - proponuje modyfikacje algorytmu i programu realizującego rozkład liczby na czynniki pierwsze - układa algorytm i program realizujący rozkład liczby na iloczyn potęg liczb pierwszych
Jak znaleźć największy wspólny dzielnik? Algorytm Euklidesa. (11)	Uczeń: - definiuje NWD - umie obliczyć NWD dla niewielkich liczb na podstawie gotowego algorytmu Euklidesa	Uczeń: - zna i omawia różnicę pomiędzy metodami Euklidesa - umie znaleźć NWD na podstawie obu zapisów metody Euklidesa - rozumie na czym	Uczeń: - umie samodzielnie zapisać algorytm Euklidesa w postaci schematu blokowego - zna metodę iteracyjną znajdowania NWD - analizuje i poprawnie	Uczeń: - samodzielnie układa algorytm dla metody iteracyjnej znajdowania NWD - samodzielnie układa program dla znanych metod znajdowania NWD	Uczeń: - układa programy realizujące znajdowanie NWD z uwzględnieniem wyświetlające ilość realizowanych obiegów pętli

		polega metoda Euklidesa	interpretuje gotowy program realizujący algorytm Euklidesa		
Ciąg Fibonacciego (12)	Uczeń: - omawia jak powstaje ciąg Fibonacciego - wie czym ogólnie różni się metoda iteracyjna od rekurencyjnej	Uczeń: - umie zapisać kilka pierwszych działań algorytmu Fibonacciego - umie sprawdzić działanie algorytmu Fibonacciego zapisanego w postaci schematu blokowego	Uczeń: - opisuje metodę iteracyjną i rekurencyjną powstawania ciągu Fibonnaciego - analizuje programy napisane na podstawie obu metod tworzenia ciągu Fibonnacciego	Uczeń: - samodzielnie zapisuje algorytmy powstawania ciągu Fibonacciego metoda iteracyjną lub rekurencyjną - układa programy na podstawie tych algorytmów	Uczeń: - podaje przykłady występowania ciągu Fibonacciego w naturze
Wydawanie reszty metoda zachłanną (13)	Uczeń: - wie na czym polega wydawanie reszty metodą zachłanną (maksymalne nominały)	Uczeń: - "wydawać resztę" w różnych nominałach na kilka sposobów omawiając poszczególne czynności w sposób algorytmiczny - analizuje algorytm wydawania reszty metoda zachłanną w postaci schematu blokowego	Uczeń: - umie zapisać algorytm metody zachłannej wydawania reszty - analizuje program ułożony wg algorytmu metody zachłannej	Uczeń: - umie ułożyć algorytm w postaci schematu blokowego dla funkcji obliczania nominałów dla reszty z odejmowaniem - modyfikuje algorytm i program zmieniając wartość nominałów	Uczeń: - samodzielnie układa program wydający resztę z groszami
Znajdowanie jednocześnie elementu najmniejszego i największego (algorytm naiwny) (14)	Uczeń: - wie na czym polega proces sortowania - zna pojęcie tablicy (macierzy) - umie własnymi słowami z pomocą nauczyciela przedstawić sposób znajdowania najmniejszego i największego elementu	Uczeń: - umie znaleźć największy i najmniejszy element tablicy działając na podstawie algorytmu naiwnego - omawia podstawy metody zastosowanej w algorytmie naiwnym	Uczeń: - umie zapisać algorytm naiwny w jednej z postaci - umie sprawdzić poprawność działania algorytmu naiwnego - analizuje program działający wg algorytmu naiwnego	Uczeń: - samodzielnie zapisuje algorytm naiwny w dowolnej postaci - układa program (procedurę) na podstawie algorytmu naiwnego - wie do jakiej klasy złożoności obliczeniowej należy algorytm naiwny	Uczeń: - oblicza ile porównań zostaje wykonanych podczas naiwnego algorytmu jednoczesnego wyznaczania min i max

	w tablicy				
Znajdowanie jednocześnie elementu największego i najmniejszego (algorytm optymalny) (15)(16)	Uczeń: - wie na czym polega metoda "dziel i zwyciężaj" - wie jakie warunki musi spełniać tablica dla metody "dziel i zwyciężaj" - zna cechy metod rekurencyjnych i nierekurencyjnych - z pomocą nauczyciela omawia gotowy algorytm zgodny z metodą "dziel i zwyciężaj" oraz optymalny	Uczeń: - umie przygotować tablicę do zastosowania metody "dziel i zwyciężaj" - samodzielnie omawia gotowy algorytm zgodny z metodą "dziel i zwyciężaj" - samodzielnie omawia gotowy algorytm zgodny z metodą optymalną	Uczeń: - samodzielnie zapisuje algorytm znajdowania jednocześnie elementu największego i najmniejszego metoda "dziel i zwyciężaj" w postaci listy kroków - analizuje program działający wg tego algorytmu - samodzielnie zapisuje algorytm znajdowania jednocześnie elementu największego i najmniejszego metodą nierekurencyjną (algorytm optymalny) w postaci listy kroków - analizuje program działający wg tego algorytmu	Uczeń: - samodzielnie układa algorytm znajdowania jednocześnie elementu największego i najmniejszego metoda "dziel i zwyciężaj" w postaci schematu blokowego - układa program na podstawie tego algorytmu - układa program na podstawie algorytmu wykorzystującego podział tablicy na 2 części i debuguje go w celu zbadania kolejności sprawdzania obu części tablic - samodzielnie układa algorytm znajdowania jednocześnie elementu największego i najmniejszego wg algorytmu optymalnego w postaci schematu blokowego i listy kroków - układa program na podstawie obu metod	Uczeń: - analizuje ilość porównań z metody rekurencyjnej i porównuje ją z metodą algorytmu naiwnego - dokonuje porównania stopnia złożoności obu algorytmów
Sortowanie bąbelkowe - przez prostą zmianę (17)	Uczeń: - rozumie ideę sortowania bąbelkowego - opisuje własnymi słowami metodę bąbelkową sortowania - sortuje z pomocą nauczyciela niewielką	Uczeń: - dokładnie wyjaśnia na przykładach metodę sortowania bąbelkowego - samodzielnie sortuje niewielką ilość liczb metodą bąbelkową na podstawie algorytmu w postaci listy kroków	Uczeń: - samodzielnie zapisuje algorytm sortowania bąbelkowego w postaci listy kroków - analizuje program realizujący algorytm sortowania bąbelkowego	Uczeń: - samodzielnie układa algorytm metody bąbelkowej dla n elementów w postaci schematu blokowego i omawia funkcje poszczególnych bloków - układa program na podstawie tego algorytmu	Uczeń: - zna i umie przedstawić podstawy działania algorytmu grzebieniowego - wskazuje analogie pomiędzy metodą bąbelkową a grzebieniową

	ilość liczb metodą bąbelkową	(podanej przez nauczyciela)			
Sortowanie przez wybieranie (18)	Uczeń: - rozumie ideę sortowania przez wybieranie - opisuje własnymi słowami metodę sortowania przez wybieranie - sortuje z pomocą nauczyciela niewielką ilość liczb metodą przez wybieranie	Uczeń: - dokładnie wyjaśnia na przykładach metodę sortowania przez wybieranie - samodzielnie sortuje niewielką ilość liczb metodą sortowania przez wybieranie na podstawie algorytmu w postaci listy kroków (podanej przez nauczyciela)	Uczeń: - samodzielnie zapisuje algorytm sortowania przez wybieranie w postaci listy kroków - analizuje program realizujący algorytm sortowania przez wybieranie	Uczeń: - samodzielnie układa algorytm metody sortowania przez wybieranie w postaci schematu blokowego i omawia funkcje poszczególnych bloków - układa program na podstawie tego algorytmu	Uczeń: - zna i umie przedstawić podstawy działania algorytmu "przez zliczanie" - wskazuje analogie pomiędzy metodą sortowania przez wybieranie a metodą "przez zliczanie" - umie oszacować z ilu pętli składa się algorytm sortowania "przez wybieranie"
Sortowanie przez wstawianie (19)	Uczeń: - rozumie ideę sortowania przez wstawianie i omawia ją przez podanie analogii do wachlarza kart - opisuje własnymi słowami metodę sortowania przez wstawianie - sortuje z pomocą nauczyciela niewielką ilość liczb metodą sortowania przez wstawianie	Uczeń: - dokładnie wyjaśnia na przykładach metodę sortowania przez wstawianie - samodzielnie sortuje niewielką ilość liczb metodą sortowania przez wstawianie na podstawie algorytmu w postaci listy kroków (podanej przez nauczyciela)	Uczeń: - samodzielnie zapisuje algorytm sortowania przez wstawianie w postaci listy kroków - analizuje program realizujący algorytm sortowania przez wstawianie	Uczeń: - samodzielnie układa algorytm metody sortowania przez wstawianie w postaci schematu blokowego i omawia funkcje poszczególnych bloków - układa program na podstawie tego algorytmu	Uczeń: - wymienia struktury danych (zamiast tablic) które można wykorzystać przy sortowaniu przez wstawianie i uzasadnia swój wybór
Sortowanie algorytmem szybkim (20)	Uczeń: - rozumie ideę sortowania algorytmem szybkim	Uczeń: - omawia znaczenie pivota w szybkim algorytmie sortowania	Uczeń: - samodzielnie zapisuje algorytm szybki sortowania w postaci listy	Uczeń: - samodzielnie układa algorytm na podstawie metody szybkiej sortowania w postaci	Uczeń: - umie udowodnić tezę, że algorytm sortowania szybki jest najszybszym algorytmem sortowania

	- opisuje własnymi słowami metodę sortowania algorytmem szybkim - sortuje z pomocą nauczyciela niewielką ilość liczb metodą sortowania algorytmem szybkim - wie czym jest pivot	- dokładnie wyjaśnia na przykładach metodę sortowania algorytmem szybkim - samodzielnie sortuje niewielką ilość liczb na podstawie algorytmu szybkiego w postaci listy kroków (podanej przez nauczyciela)	kroków - analizuje program realizujący szybki algorytm sortowania - uzasadnia dlaczego szybki algorytm sortowania jest sortowanie metodą dziel i zwyciężaj	schematu blokowego i omawia funkcje poszczególnych bloków - układa program na podstawie tego algorytmu - określa złożoność obliczeniową algorytmu metody szybkiej sortowania	
Sortowanie przez scalanie (21)	Uczeń: - rozumie potrzebę budowania dodatkowej tablicy - rozumie ideę sortowania przez scalanie - opisuje własnymi słowami metodę sortowania przez scalanie - sortuje z pomocą nauczyciela niewielką ilość liczb metodą sortowania przez scalanie	Uczeń: - omawia znaczenie dodatkowej tablicy w algorytmie sortowania przez scalanie - dokładnie wyjaśnia na przykładach metodę sortowania przez scalanie - samodzielnie sortuje niewielką ilość liczb na podstawie metody sortowania przez scalanie w postaci listy kroków (podanej przez nauczyciela)	Uczeń: - samodzielnie zapisuje algorytm sortowania przez scalanie w postaci listy kroków - analizuje program realizujący algorytm sortowania przez scalanie	Uczeń: - samodzielnie układa algorytm na podstawie metody sortowania przez scalanie w postaci schematu blokowego i omawia funkcje poszczególnych bloków - układa program na podstawie tego algorytmu - określa złożoność obliczeniową algorytmu sortowania przez scalanie	Uczeń: - umie udowodnić tezę, że algorytm sortowania przez scalanie jest zbudowany w oparciu o metodę dziel i zwyciężaj
Sortowanie kubełkowe (22)	Uczeń: - rozumie ideę sortowania kubełkowego - wie czym jest lista porządkowa - opisuje własnymi słowami metodę	Uczeń: - omawia znaczenie dzielenia danych na kubełki w algorytmie sortowania kubełkowego - dokładnie wyjaśnia na przykładach metodę sortowania kubełkowego	Uczeń: - samodzielnie zapisuje algorytm sortowania kubełkowego w postaci listy kroków - analizuje program realizujący algorytm sortowania metodą kubełkową	Uczeń: - samodzielnie układa algorytm na podstawie metody sortowania kubełkowego w postaci schematu blokowego i omawia funkcje poszczególnych bloków - określa złożoność	Uczeń: - określa złożoność obliczeniową algorytmu sortowania kubełkowego i uzasadnia swoją tezę - układa program na podstawie algorytmu sortowania kubełkowego

	sortowania kubełkowego - sortuje z pomocą nauczyciela niewielką ilość liczb metodą sortowania kubełkowego	- samodzielnie sortuje niewielką ilość liczb na podstawie metody sortowania kubełkowego w postaci listy kroków (podanej przez nauczyciela)		obliczeniową algorytmu sortowania kubełkowego - omawia przypadki w których stosuje się metodą sortowania kubełkowego	
Algorytmy numeryczne					
Wyznaczanie przybliżonej wartości pierwiastka kwadratowego metodą Newtona-Raphsona (23)(24)	Uczeń: - wymienia funkcje z języka programowania realizujące kwadrat liczby oraz pierwiastek kwadratowy - umie zapisac z pamięci ogólny wzór na metodę Newtona-Raphsona	Uczeń: - umie opisać metodę Newtona-Raphsona na podstawie algorytmu podanego metoda graficzną (prostokąty) - na podstawie opisu graficznego metody Newtona-Raphsona potrafi obliczyć pierwiastek kwadratowy niewielkie liczby - rozumie znaczenie epsilonu	Uczeń: - samodzielnie przedstawia graficznie metodę Newtona-Raphsona - samodzielnie omawia metodę Newtona-Raphsona - wie od czego zależy epsilon - zna tę metodę także pod nazwą algorytmu Herona	Uczeń: - samodzielnie przedstawia metodę Newtona-Raphsona w postaci algorytmu zapisanego wzorem w postaci schematu blokowego	Uczeń: - układa własną funkcję w języku programowania zastępującą funkcje wbudowaną np. sqrt(x)
Obliczanie wartości wielomianu (schemat Hornera) (25)(26)	Uczeń: - biegle oblicza wartość wielomianu dla danej wartości zmiennej metodą algebraiczną - zna pojęcie wielomianu - zna zalety metody Hornera	Uczeń: - działania metody algebraicznej obliczania wartości wielomianu w notacji języka programowania np. FreePascala - wie jakie korzyści przyniesie wykorzystanie do obliczeń schematu Hornera	Uczeń: - oblicza wartość wielomianu dla danej zmiennej korzystając ze schematu Hornera - uzasadnia zalety tej metody - samodzielnie omawia działanie algorytmu podanego w postaci schematu blokowego lub listy kroków	Uczeń: - samodzielnie zapisuje algorytm korzystający ze schematu Hornera w postaci listy kroków i schematu blokowego - na ich podstawie dokładnie wyjaśnia metodę Hornera i uzasadnia jej zalety - samodzielnie zapisuje uogólnioną postać schematu Hornera	Uczeń: - samodzielnie układa funkcję realizującą obliczanie wartości wielomianu metodą Hornera

			- omawia uogólnioną postać schematu Hornera		
Prezentowanie liczb w różnych systemach liczbowych (27)	Uczeń: - zna podstawowe systemy zapisu liczb - wie na jakiej zasadzie tworzy się liczbowe systemy wagowe - z pomoca nauczyciela umie konwertować liczby NKB, HEX i dziesiętne	Uczeń: - rozumie rozwiązanie zadania z rozdziału 24 - umie zbudować system wagowy o dowolnej podstawie tłumacząc zasadę jego powstawania - umie przedstawiać liczby w podstawowych systemach liczbowych	Uczeń: - omawia działanie algorytmu zamiany liczby na dowolny kod wagowy - korzystając z gotowego algorytmu przedstawia liczby w różnych systemach - zna wzór wynikający ze schematu Hornera	Uczeń: - samodzielnie układa algorytm w postaci schematu blokowego lub listy kroków zmieniający prezentacje danej liczby w różnych systemach liczbowych - układa program realizujący ten algorytm - rozumie budowę i znaczenie wzoru wynikającego ze schematy Hornera	Uczeń: - tłumaczy zastosowanie schematu Hornera do prezentacji liczb w różnych systemach
Szybkie podnoszenie do potęgi (schemat Hornera "od lewej do prawej") (28)(29)	Uczeń: - rozumie wzór podnoszący liczbę do potęgi dla konkretnego przykładu z wykorzystaniem schematu Hornera - rozumie różnice stosowania klasycznej metody wielokrotnego mnożenia w czasie potęgowania, a metody "od lewej do prawej"	Uczeń: - oblicza potęgę liczby wg wzoru wynikającego ze schematu Hornera dla metody "od lewej do prawej" - objaśnia metodę "od lewej do prawej" - rozumie znaczenie postaci liczby binarnej dla powstania metody "od lewej do prawej"	Uczeń: - samodzielnie zapisuje działania, które należy wykonać wg metody na obliczanie potęgi "od lewej do prawej" podczas operacji potęgowania liczby - rozumie działanie algorytmu potęgowania metodą "od lewej do prawej"	Uczeń: - samodzielnie zapisuje algorytmy wg metody "od lewej do prawej" w postaci schematu blokowego i listy kroków - umie ustalić liczbę mnożeń podczas realizacji algorytmu obliczania potęgi metodą "od lewej do prawej" - samodzielnie układa funkcję realizującą podnoszenie liczby do potęgi	Uczeń: - zna i umie zastosować metodę rekurencyjną szybkiego podnoszenia do potęgi
Wyznaczanie miejsc zerowych funkcji (30)	Uczeń: - zna i rozumie definicję miejsca zerowego funkcji - zna i rozumie definicję funkcji liniowej, kwadratowej i sześciennej	Uczeń: - wymienia i rozumie założenia i warunki dla zastosowania metody bisekcji(połowienia) - rozumie algorytm metody bisekcji (połowienia) zapisany w	Uczeń: - oblicza miejsca zerowe funkcji na podstawie podanego algorytmu dla metody bisekcji (połowienia) - ilustruje te czynności na wykresie	Uczeń: - samodzielnie układa algorytm dla metody bisekcji (połowienia) - zna twierdzenie Bolzano-Couchy`ego - układa program realizujący algorytm metody bisekcji	Uczeń: - samodzielnie wyznacza Epsilon dla danego algorytmu wg. metody bisekcji (połowienia)

	- zna pojęcie funkcji wielomianowej - umie sporządzać i odczytywać wykresy funkcji na płaszczyźnie - zna i rozumie pojęcie dziedziny funkcji - umie wskazać miejsca zerowe na wykresie funkcji	postaci listy kroków		(połowienia)	
Obliczanie pola obszarów zamkniętych (31)(32)	Uczeń: - rozumie pojęcie obszaru zamkniętego - rozumie pojęcie "przybliżona metoda obliczeń" - z pomocą nauczyciela omawia metodę prostokątów	Uczeń: - wie na czym polegają metody przybliżonego obliczania pola obszaru zamkniętego - wskazuje różnice pomiędzy metoda prostokątów, a trapezów - uzasadnia dokładność poszczególnych metod	Uczeń: - samodzielnie omawia metodę trapezów na podstawie przykładowego wykresu - samodzielnie omawia metodę prostokątów na podstawie przykładowego wykresu - omawia wpływ gęstości podziału pola na dokładność obliczenia pola	Uczeń: - samodzielnie zapisuje wzory dla metody prostokątów - samodzielnie zapisuje wzory dla metody trapezów - ocenia błąd poszczególnych metod - zapisuje i rozumie wzór trapezów uwzględniający błąd obliczeń - układa program realizujący obliczanie pola obszaru zamkniętego metodą prostokątów - układa program realizujący obliczanie pola obszaru zamkniętego metodą trapezów	Uczeń: - zna pojęcie całkowania numerycznego - zapisuje i rozumie algorytm dla metody Simpsona
Algorytmy cz.2.					
Sprawdzanie ciągu znaków na występowanie palindromu (33)	Uczeń: - wie czym jest palindrom i jakie ma właściwości	Uczeń: - omawia metodę na podstawie algorytmu podanego przez	Uczeń: - omawia zachowanie się algorytmu w przypadkach parzystych i nieparzystych ilości znaków w ciągu	Uczeń: - samodzielnie zapisuje algorytm w postaci schematu blokowego - samodzielnie układa	Uczeń: Układa algorytm wyszukujący palindromy w ciągu znaków niebędącym w całości palindromem

	- podaje kilka przykładów palindromów - z pomocą nauczyciela omawia najprostszą metodę sprawdzania czy dany ciąg znaków jest palindromem	nauczyciela -	- analizuje program realizujący wyszukiwanie palindromów	program wyszukujący palindromy w ciągu znaków	
Wyszukiwanie anagramów dany ciąg znaków tworzy anagram (34)	Uczeń: - omawia różnice pomiędzy palindromem a anagramem - podaje przykłady anagramów - umie oszacować czy dany ciąg jest anagramem	Uczeń: - samodzielnie sprawdza czy słowa są anagramami - własnymi słowami przedstawia metodę wyszukiwania anagramów	Uczeń: - układa algorytm w postaci schematu blokowego sprawdzający czy ciąg znaków jest anagramem - układa algorytm w postaci listy kroków sprawdzający czy ciąg znaków jest anagramem	Uczeń: - samodzielnie układa program sprawdzający czy ciąg znaków jest anagramem uwzględniając przestawianie liter	Uczeń: - samodzielnie układa program sprawdzający czy ciąg znaków jest anagramem uwzględniając przestawianie liter i sylab
Alfabetyczne sortowanie wyrazów (35)	Uczeń: - wie na czym polega sortowanie alfabetyczne i gdzie znajduje zastosowanie w informatyce - z pomocą nauczyciela analizuje przedstawiony algorytm (np. z podręcznika)	Uczeń: - wyjaśnia pojęcie "porządkowanie leksykograficzne" - samodzielnie podaje przykłady takiego sortowania - wyjaśnia działanie algorytmu w postaci listy kroków porządkującego ciągi cyfr - wie, że liczba etapów sortowania leksykograficznego jest równa liczbie znaków najdłuższego, badanego	Uczeń: - samodzielnie przedstawia sposób sortowania leksykograficznego kubełkowego - wyjaśnia na przykładzie zasadę działania tego algorytmu	Uczeń: - układa program realizujący algorytm sortowania leksykograficznego ciągów znaków składających się z cyfr metodą kubełkową	Uczeń: - układa program realizujący algorytm sortowania leksykograficznego dowolnych ciągów znaków metodą kubełkową

		łańcucha znaków			
Wyszukiwanie wzorca w danym teście. Metoda naiwna (36)	Uczeń: - wyjaśnia znaczenie wyszukiwania wzorców w tekście - samodzielnie, stosując metodę naiwną, wyszukuje wzorzec w przykładowym ciągu znaków	Uczeń: - wyjaśnia zasady algorytmu naiwnego wyszukiwania wzorców w ciągu znaków - omawia gotowy algorytm wyszukiwania metoda naiwną	Uczeń: - układa algorytm w postaci listy kroków, realizujący metoda naiwną wyszukiwania wzorca w ciągu znaków - układa program wyszukujący metoda naiwną wzorzec w danym ciągu znaków	Uczeń: - układa algorytm w postaci schematu blokowego, realizujący metoda naiwną wyszukiwania wzorca w ciągu znaków - układa program wyszukujący metoda naiwną wzorzec w danym ciągu znaków zmodyfikowany o pomijanie zbędnych wyszukiwań	Uczeń: - układa program realizujący wyszukiwanie kilku wzorców w tym samym ciągu
Wyszukiwanie wzorca w danym teście. Metoda Boyera-Moore`a (37)	Uczeń: - umie wymienić niedoskonałości metody naiwnej - z pomocą nauczyciela wymienia zalety metody Boyera-Moore`a	Uczeń: - wyjaśnia sposób wyszukiwania metodą Boyera-Moore`a na podstawie przykładu przedstawionego w podręczniku	Uczeń: - układa algorytm wyszukiwania wzorca metodą Boyera-Moore`a (w postaci listy kroków) - omawia jego działanie na przykładzie	Uczeń: - układa algorytm wyszukiwania wzorca metodą Boyera-Moore`a (w postaci schematu blokowego) - układa program realizujący algorytm Boyera-Moore`a	Uczeń: - układa program realizujący algorytm Boyera-Moore`a pobierający z pliku tekstowego ciągi znaków i wzorce
Obliczanie wartości wyrażenia podanego w postaci ONP (38) (39)	Uczeń: - zna różnicę pomiędzy przekształceniami wyrażenia arytmetycznego metodą ONP a klasyczną notacją z nawiasami - z pomocą nauczyciela wyjaśnia na czym polega metoda ONP	Uczeń: - wyjaśnia metodę ONP przyrostową na podstawie opisu graficznego - wyjaśnia metodę ONP przedrostkową na podstawie opisu graficznego	Uczeń: - przedstawia w postaci graficznej przekształcenie danego wyrażenia arytmetycznego metodą NP przedrostkową - przedstawia w postaci graficznej przekształcenie danego wyrażenia arytmetycznego metodą NP przyrostkową	Uczeń: - układa algorytm przekształcania wyrażeń arytmetycznych metodą ONP - układa program realizujący ten algorytm	Uczeń: - uzupełnia algorytm i program przekształcający wyrażenie arytmetyczne o procedurę obliczającą jego wartość dla różnych wartości poszczególnych argumentów
Algorytmy kompresji i szyfrowania					
Szyfr Cezara, szyfr przestawieniowy (40)	Uczeń: - zna zastosowanie szyfrowania w	Uczeń: - samodzielnie szyfruje ciąg znaków metoda	Uczeń: - zapisuje algorytm Cezara w postaci listy	Uczeń: - zapisuje algorytm Cezara w postaci	Uczeń: - układa program, który szyfruje lub odszyfrowuje

	informatyce - z pomocą nauczyciela szyfruje ciąg znaków metoda Cezara	Cezara - szyfruje ciąg znaków metodą pasków na podstawie opisu - szyfruje ciąg znaków metodą szyfru podstawieniowego wieloalfabetowego na podstawie opisu - szyfruje ciąg znaków metodą szyfru przestawieniowego na podstawie opisu	kroków - zapisuje algorytm oparty na metodzie pasków w postaci listy kroków - zapisuje algorytm realizujący metodę szyfru podstawieniowego wieloalfabetowego w postaci listy kroków - zapisuje algorytm realizujący metodę szyfru przestawieniowego w postaci listy kroków	schematu blokowego - zapisuje algorytm oparty na metodzie pasków w postaci schematu blokowego - zapisuje algorytm realizujący metodę szyfru podstawieniowego wieloalfabetowego w postaci schematu blokowego - zapisuje algorytm realizujący metodę szyfru przestawieniowego w postaci schematu blokowego - układa program na podstawie algorytmów szyfru Ceazara, podstawieniowego wieloalfabetowego, Przestawieniowego, pasków	ciąg znaków pobrany z pliku tekstowego
Kody znaków o zmiennej długości. Alfabet Morse`a (41)	Uczeń: - zna zastosowanie alfabetu Morse`a - zna cechy alfabetu Morse`a	Uczeń: - koduje tekst w alfabecie Morse`a korzystając z tabeli kodu z literami, cyframi i znakami specjalnymi - rozumie działanie algorytmu kodowania w Alfabecie Morse`a	Uczeń: - układa algorytm kodowania Alfabetem MOrse`a i zapisuje go w postaci listy kroków lub schematu blokowego	Uczeń: - układa program komputerowy na podstawie algorytmu kodowania Alfabetu Morse`a wyświetlający zakodowany tekst na ekranie	Uczeń: - modyfikuje program w taki sposób, by zakodowany tekst oprócz wyświetlania na ekranie był "nadawany" sygnałami dźwiękowymi lub sygnalizacja rozbłyskujących punktów na ekranie (symulacja nadawania światłem)
Kody Huffmana (42)(43)	Uczeń: - wie jak można	Uczeń: - omawia metodę kodów	Uczeń: - układa algorytm tworzenia	Uczeń: - układa program na	Uczeń: - układa program na

	zakodować np. 5-znakowy alfabet za pomocą 3 bitów kodu binarnego - wie czym jest drzewo Huffmana - z pomocą nauczyciela wyjaśnia podstawy kodu Huffmana - uzasadnia zasadę kodowania najczęściej używanych słów najkrótszym ciągiem znaków	Huffmana na podstawie przykładu np. z podręcznika - zna budowę drzewa Huffmana	słów kodowych dla kody Huffmana w postaci listy kroków - układa algorytm tworzenia słów kodowych dla kody Huffmana w postaci schematu blokowego (dla alfabetu 5-znakowego)	podstawie algorytmu kodowania kodem Huffmana (dla alfabetu 5-znakowego)	podstawie algorytmu kodowania kodem Huffmana dla alfabetu 7-znakowego
Szyfr z kluczem publicznym (44)(45)	Uczeń: - rozumie ideę szyfrowania z kluczem - tłumaczy na podstawie gotowego rysunku (np. z podręcznika) ideę szyfrowania z kluczem symetrycznym - tłumaczy na podstawie gotowego rysunku (np. z podręcznika) ideę szyfrowania z kluczem asymetrycznym - rozumie pojęcie kongruencji	Uczeń: - samodzielnie omawia na podstawie własnego rysunku ideę szyfrowania z kluczem symetrycznym - samodzielnie omawia na podstawie własnego rysunku ideę szyfrowania z kluczem symetrycznym - samodzielnie omawia na podstawie własnego rysunku ideę szyfrowania z kluczem asymetrycznym	Uczeń: - na podstawie algorytmu z podręcznika tłumaczy metodę powstawania kluczy prywatnych - na podstawie algorytmu z podręcznika tłumaczy metodę powstawania kluczy publicznych	Uczeń: - samodzielnie układa algorytm tworzenia kluczy prywatnych - samodzielnie układa algorytm tworzenia kluczy publicznych - samodzielnie układa algorytm szyfrowania i deszyfrowania metodą RSA	Uczeń: - układa program wykonujący szyfrowanie i deszyfrowanie metodą RSA
Algorytmy badające własności geometryczne					
Badanie warunków trójkąta (46)	Uczeń: - rozumie pojęcie warunek trójkąta	Uczeń: - samodzielnie analizuje i omawia algorytm	Uczeń: - samodzielnie układa algorytm badania	Uczeń: - układa program badający warunek trójką na podstawie algorytmu	Uczeń: - modyfikuje program ułożony na podstawie algorytmu zabezpieczając

	- umie zapisać równania twierdzenie cosinusów dla każdego z boków trójkąta - z pomocą nauczyciela analizuje algorytm badania warunku trójkąta w postaci listy kroków	badania warunku trójkąta w postaci listy kroków	warunku trójkąta w postaci listy kroków i omawia jego działanie		go przed wprowadzaniem nieprawidłowych danych
Badanie położenia punktów względem prostej (47)	Uczeń: - posługuje się kartezjańskim układem współrzędnych i umie wyznaczać położenie punktów - wie czym jest odległość punktu od prostej i jak się go wyznacza	Uczeń: - samodzielnie analizuje i omawia algorytm badania położenia punktów względem prostej w postaci listy kroków - tłumaczy jego działanie posługując sie wykresami na płaszczyźnie	Uczeń: - samodzielnie układa algorytm badania położenia punktów względem prostej - analizuje gotowy program ułożony na podstawie algorytmu badania położenia punktów względem prostej	Uczeń: - układa program na podstawie algorytmu badania położenia punktów względem prostej	Uczeń: - modyfikuje program w taki sposób, by możliwe było wpisywanie danych z klawiatury - modyfikuje program w taki sposób, by sygnalizował po której stronie prostej znajdują się badane punkty
Badanie przynależności punktu do odcinka (48)	Uczeń: - wie jakie są warunki przynależności punktu do odcinka - zna wzór do obliczania odległości punktów w kartezjańskim układzie współrzędnych - zna pojęcie nachylenia odcinka - z pomocą nauczyciela podany algorytm do badania przynależności punktu do odcinka	Uczeń: - samodzielnie tłumaczy pojęcie przynależności punktu do odcinka - samodzielnie tłumaczy pojęcie odległości punktów w kartezjańskim układzie współrzędnych - samodzielnie tłumaczy pojęcie nachylenia odcinka - wymienia i omawia warunki przynależności punktu do odcinka	Uczeń: - układa algorytm badania przynależności punktu do odcinka - analizuje gotowy program realizujący algorytm badania przynależności punktu do odcinka	Uczeń: - układa program na podstawie algorytmu badania przynależności punktu do odcinka dla kilku przypadków	Uczeń: - układa program na podstawie algorytmu badania przynależności punktu do odcinka dla wszystkich możliwych przypadków
Badanie przecinania	Uczeń:	Uczeń:	Uczeń:	Uczeń:	Uczeń:

się odcinków (49)	- wie jakie warunki muszą być spełnione by odcinki się przecinały lub nie	- rozumie metodę wyznaczania względnego położenia trzech punktów - na podstawie gotowego opisu tłumaczy metodę badania przecinania się odcinków na płaszczyźnie kartezjańskiej	- układa algorytm badania przecinania się odcinków w postaci schematu blokowego - analizuje gotowy program (np. z podręcznika) realizujący układa algorytm badania przecinania się odcinków	- układa program realizujący układa algorytm badania przecinania się odcinków	- rozumie i omawia metodę "zamiatania płaszczyzny"
Badanie przynależności punktu do wielokąta (50)(51)	Uczeń: - zna i rozumie definicję wielokąta na płaszczyźnie - wie czym jest półprosta - rozumie kryteria przynależności punktu do wielokąta przedstawione np. w podręczniku	Uczeń: - wymienia kryteria przynależności punktu do wielokąta - omawia metodę badania przynależności punktu do wielokąta - analizuje gotowy program realizujący algorytm badania przynależności punktu do wielokąta (np. z podręcznika)	Uczeń: - układa algorytm w postaci listy kroków realizujący badanie przynależności punktu do wielokąta - układa algorytm w postaci schematu blokowego realizujący badanie przynależności punktu do wielokąta - analizuje program (np. z podręcznika) realizujący algorytm badania przynależności punktu do wielokąta	Uczeń: - układa program realizujący algorytm badania przynależności punktu do wielokąta	Uczeń: - układa program realizujący algorytm badania przynależności punktu do obszaru ograniczonego bokami wielokąta
Dywan Sierpińskiego (52) (53)	Uczeń: - zna pojęcie rekurencji - wie jak wygląda trójkątny i kwadratowy dywan Sierpińskiego - zna pojęcie „fraktal” i kojarzy z nim dywan Sierpińskiego	Uczeń: - wyjaśnia w jaki sposób powstaje fraktal dywan Sierpińskiego - przedstawia w sposób graficzny powstawanie dywanu Sierpińskiego - rozumie działanie algorytmu powstawania dywanu Sierpińskiego	Uczeń: - układa algorytmy rysowania trójkątnego i kwadratowego dywanu Sierpińskiego	Uczeń: - układa program tworzący na ekranie dywan Sierpińskiego w zadanej ilości kroków	Uczeń: - rozbudowuje program o możliwość wprowadzania ilości kroków
Płatek (śnieżynka) Kocha. Drzewo	Uczeń: - rozpoznaje na	Uczeń: - opisuje proces	Uczeń: - układa algorytm tworzenia	Uczeń: - układa program rysujący	Uczeń: - ocenia czas rysowania

binarne. (54)(55)	rysunku płatek Kocha - z pomocą nauczyciela opisuje proces powstawania śnieżynki Kocha	powstawania śnieżynki kocha za pomocą listy kroków - zna i opisuje na przykładzie proces powstawania drzewa binarnego	śnieżynki Kocha za pomocą schematu blokowego - analizuje program rysujący fraktal śnieżynka Kocha - analizuje program rysujący fraktal drzewo binarne	płatek Kocha o stałym, niewielkim stopniu - układa program rysujący drzewo binarne	płatka Kocha o zadanym stopniu
Realizacja projektu informatycznego					
Wydobywanie wymagań klienta i ich specyfikacja (56)	Uczeń: - umie rozumie znaczenie poprawnej komunikacji z klientem - rozumie pojęcie „wydobyć wymagania” - umie zapisać wymagania klienta i zaproponować modyfikację tej listy - przeprowadził ćwiczenia z ustalania wymagań klienta - wie jakie znaczenie ma rozmowa z klientem i poznanie środowiska w jakim ma działać program - zna pojęcie specyfikacji wymagań i umie ją sporządzić na podstawie wydobycia wymagań klienta - jest świadomy kosztów błędów popełnionych na etapie tworzenia specyfikacji wymagań - umie zapisać specyfikację wymagań w postaci tabeli z odpowiednimi danymi – cel, dostawca, odbiorca …				
Tworzenie dokumentacji projektu (57)	Uczeń: - wie jakie znaczenie ma tworzenie dobrej dokumentacji projektu - zna strukturę dokumentu projektu - umie wymienić elementy dokumentu - zna strukturę tabeli dokumentu projektu - układa dokument projektu dla konkretnego przykładu i listy wymagań - przeprowadza weryfikację swojego dokumentu projektu				
Implementacja i kodowanie programu (58)	Uczeń: - wie jakie znaczenie ma odpowiednie planowanie prac nad programem komputerowym - zna zasady ustalania standardów obowiązujących programistów w zespole - wie na czym polegają testy jednostkowe i integracyjne - wie jakie kwalifikacje powinien mieć weryfikator kodu - wie na cym polega wersjonowanie źródeł - umie sporządzić dokument w którym zostaną spisane ustalenia programistów (nazwy zmiennych itp.) - zna pojęcie repozytorium - wie jakie znaczenie ma uporządkowanie archiwum źródeł dla pracy zespołu programistów realizujących projekt				
Testowanie	Uczeń:				

programu (59)	- jest świadomy ważności testowania programów różnych okresach i na różnych poziomach ich powstawania - wie jakie znaczenie ma jak najszybsze wykrycie błędów i jakie mogą być konsekwencje ich późnego wykrycia - wie na czym polega proces testowania program na różnych poziomach jego powstawania - zna pojęcie przypadku i scenariusza testowego - umie ułożyć scenariusz testowy dla przykładowego program - wie, ze w przypadku wykrycia błędu należy zbadać czy błąd się powtarza i w jakich warunkach powstaje - umie sporządzić dokument zgłoszenia błedu
Sporządzanie dokumentacji technicznej programu (60)	Uczeń: - wie co powinna zawierać dokumentacja techniczna program - wie, że dokumentacja techniczna powinna powstawać równolegle z procesem powstawania programu - umie przygotować dokumentacje techniczną dla prostego programu np. obliczającego głosy w wyborach Samorządu Uczniowskiego - sporządza opis do modułów program i wie co powinien zawierać
Sporządzanie dokumentacji użytkownika i przekazanie jej klientowi (61)	Uczeń: - wie co powinna zawierać dokumentacja użytkownika - wie w jakim celu tworzy się dokumentacje użytkownika - umie napisać instrukcję obsługi i instalacji programu - sporządza dokumentację użytkownika dla jednego z ułożonych wcześniej programów zawierający spis treści np. obliczającego głosy w wyborach Samorządu Uczniowskiego - wie jakie elementy powinna zawierać dokumentacja użytkownika w przypadku bardziej skomplikowanej struktury programu np. wymagającej administracji - wie jak przebiegać powinna instalacja programu i klienta oraz testowanie i prezentacja - wie na czym polegają testy akceptacyjne - umie na podstawie testów określić czy program spełnia wszystkie założenia i oczekiwania klienta

UWAGA! Do uzyskania oceny celującej w niektórych tematach wymagane są informacje i umiejętności wykraczające poza treść podręcznika. Dla każdej oceny wymagane jest spełnienie warunków na ocenę niższą