Zad. 1 Rekurencyjny algorytm zmieniający liczbę binarną na dziesiętną + złożoność obliczeniowa.

Załóżmy, że liczba jest postaci:

```
int bin[N];

int func ( int *bin, int size, int poz )
{
	if ( poz == 0 ) return poz;

	int pt = pow ( 2, size – poz );
	return ( bin[i]*pt + func ( bin, size, poz-1 );
}
```

wywołujemy ją tak:

```
int dec = func ( bin, N, N );
```

Zad. 2 Czy n liczb całkowitych z przedziału [-2,2] można posortować w czasie O(n) + wyjaśnienie.

Tak. Możemy zastosować do tego algorytm sortowania przez zliczanie. Tworzymy 5 zmiennych przechowujących ilość wystąpień -2, -1, 0, 1 i 2.

Zliczamy ilość wystąpień danej liczby w ciągu – ilość kroków n.
Wyliczamy pozycje od których powinniśmy wstawiać dane elementy – ilość kroków 5.
Wstawiamy elementy wg. wyżej policzonych pozycji – ilość kroków n.

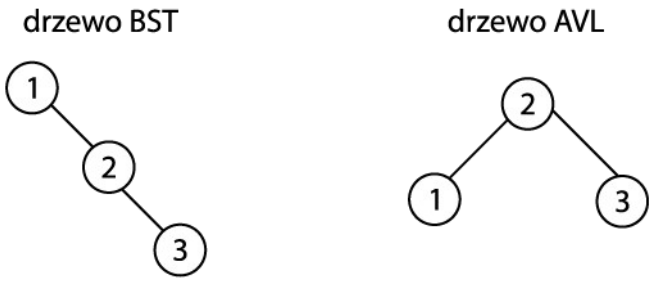
Całkowity koszt K = 2n + 5 = O (n).

Zad. 3 Zbudować drzewo BST z liczb ..... i wypisać w porządkach inorder i postorder.

..

Zad. 4 Czy drzewo z zadania 3 będzie identyczne z drzewem AVL zbudowanym z tych samych liczb.

Nie. Drzewo AVL jest wyposażone dodatkowo w algorytm wyrównujący. Przykładowo dla liczb 1, 2, 3:



Zad. 5 Algorytm odwracania niecyklicznej jednokierunkowej listy w miejscu.

```
Node *head2 = NULL;

while ( head != NULL )
{
	push ( head2, head );		//dodajemy 1 element z listy1 na początek lisy 2
	pop ( head);		//usuwamy 1 element z listy 1
}

push ( head, tmp )
{
	if ( tmp == NULL ) return;

	tmp->next = head;
	head = tmp;
}

pop ( head )
{
	if ( head == NULL ) return;

	Node *pom = head;
	head = head->next;
	delete pom;
}
```