Lecture 1 — Groups

Dr. D. Bernhard

In this lecture: groups (axioms and examples) — the group $\mathbb{Z}_n$ — order of a group — division with remainder — equivalence relations — computing in $\mathbb{Z}_n$.

Learning outcomes: After this lecture and revision, you should be able to

- Understand what a group is and check for simple examples whether an example is a group or not.
- Add and invert in the group $\mathbb{Z}_n$ for any integer $n > 0$.
- Divide integers with remainder.
- Understand the C $+$, $-$, / and % operators on unsigned integer datatypes.
- Understand what an equivalence relation is and check for simple examples whether or not a relation is an equivalence relation.

$\diamond$ More advanced material that may help interested students to further understand the topics but can safely be skipped the first few times you read the notes is marked by the $\diamond$ symbol and set in a smaller font, just like this.

Note on exercises. All exercises are graded with one to three stars according to the following scheme.

$\star$ A simple comprehension exercise. If you have understood the topic that this question is about, you should be able to answer it in a matter of minutes or even less than that.

$\star\star$ A standard exercise, may require a bit of thinking but should not take hours to solve if you've understood what it's about. At the end of the course and after revision, you should be able to solve two star exercises without too much trouble.

$\star\star\star$ An advanced exercise. Three star exercises can require a lot of work or a deeper understanding of the mathematical principles behind an idea. Leave these for last. You are not meant to solve all of these, nor will you be able to unless you have a lot of spare time.

Before we start: in this course, the natural numbers $\mathbb{N}$ are the numbers $0, 1, 2, 3, \ldots$ and the integers $\mathbb{Z}$ are the natural numbers together with the negative whole numbers: $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$

1 Groups

The topic of today's lecture is the mathematical structure called a group and the construction of a particular kind of group called $\mathbb{Z}_n$.

1.1 Motivation

A group is something "a bit like adding numbers" — there are elements which are a bit like numbers and there is something you can do to them that is a bit like adding.

There are a lot of things "like numbers", for example:

- Numbers, including variations such as integers, rational numbers, real numbers, complex numbers etc.
- Integer datatypes as used in most programming languages with a fixed bitlength. Actually, we'll mostly be sticking to unsigned integers in this course.
- Pairs, triples and other tuples or vectors of any of the above kinds of numbers.

Let's recall some things we know about addition:

1. You can add numbers in any order and get the same result. If you're asked to mentally add $5 + 3 + 7$, one way to do this if you know that $3 + 7 = 10$ is can do the sum as $5 + (3 + 7) = 5 + 10 = 15$.
2. You can cancel additions on both sides of equations. If you know that $x + 5 = 10 + 5$, you can write $x \cancel{+5} = 10 \cancel{+5}$ to get $x = 10$.

Anyone writing an optimizing compiler will be spending a lot of time on such rules to turn arithmetic expressions in programs into efficient machine code. But there is a big difference between numbers as we know them and numbers as most languages' int(eger) types use them: computer integers have a fixed size, for example 64 bits, and integers can "wrap around": by computing $1 + 1 + \ldots$, you eventually end up at 0 again.

Let's look at 8-bit unsigned integers. The smallest such value is 0 and the largest $2^8 - 1 = 255$, represented by binary $1111\ 1111$. There are many differences to normal integers. For example, in 8-bit integers it does not make sense to speak of "positive" and "negative" numbers: $128 + 128 = 0$ so we can't call 128 "positive" or "negative" according to the usual rules (the sum of positive/negative numbers is again positive/negative). Multiplying 8-bit integers is even less like working with the usual integers. For example, $16 \cdot 16 = 0$ so the product of two non-zero values can become zero. Also, $16 \cdot 32 = 0$ so we can't cancel in multiplications any more: $16 \cdot x = 16 \cdot y$ does not imply $x = y$. Do the two rules we gave above still hold for 8-bit integers? For example, if we know $16 + x = 16 + y$ in 8-bit integers, can we still conclude that $x = y$? It turns out that we can. Mathematically, this difference between adding and multiplying in 8-bit integers can be expressed by saying that for the usual integers, both addition

and multiplication (excluding 0) are group operations; for the 8-bit integers addition is a group operation but multiplication is not (whether or not you exclude 0).

A quick outlook on where we're going in the Algebra section of this course: our big question will be, can we construct operations on the set of 8-bit unsigned integers (or any other bitlength for that matter) such that we can add, subtract, multiply and divide (except by 0) according to more or less the usual rules? It will turn out that we can and that there is "essentially" only one way to do this. What "essentially" means here we will also investigate.

1.2 Definition of a group

Half of learning Algebra is deciphering the notation. We give the definition of a group and an alternative "programmer's notation" version.

Definition 1.1 (group). A group $\mathbb{G} = (G, +)$ consists of a set G and an operation $+ : G \times G \to G$ that has the following properties.

associative For any elements g, h, k of G we have $(g + h) + k = g + (h + k)$.

neutral element There is an element e of G with the property that for all elements g of G we have $e + g = g$ and $g + e = g$. We call such an element a neutral element of the group $(G, +)$.

inverses For any element g of G there is an element h of G such that $g + h = e$ and $h + g = e$ where e is the neutral element. We call h the inverse of g and write $h = (-g)$.

An alternative definition of a group is a datatype or "class" `G` with an equality operation[1] `==` and a function with signature (in C style notation) `G add(Ga, G b)` with the following properties.

associative For any elements `g, h, k` of `G` we have `add(add(g, h), k) == add(g, add(h, k))`.

neutral element There is a function `G neutral(void)` with the property that for all `g` of type `G` we have `g == add(g, neutral())` and `g == add(neutral(), g)`.

inverses There is a function `G invert(G x)` such that for any element `g` of type `G` we have `add(g, invert(g)) == neutral()` and `add(invert(g), g) == neutral()`.

[1] For the usual equality operation that we are used to this "just works". The actual condition is that the equality operation must be an equivalence relation, but this requires a bit of formalism to do correctly which we'll sweep under the carpet for now.

In this definition it is important that the functions `add`, `neutral` and `invert` are functions in the mathematical sense and not "procedures", i.e. they maintain no state between calls and do not have access to an environment such as a source of randomness. Whether you write the group operation as a function `add` or an infix operator $+$ is a matter of notation and does not change whether or not something forms a group.

These definitions immediately give the following rules for working in groups. (They are not hard to prove but we leave this as an exercise.)

Proposition 1.2. In a group:

1. There can only be one neutral element. If two elements in a group both have the properties of a neutral element, then they are equal.
2. An element can only have one inverse. If $g + h = e$ and $g + k = e$ then $h = k$. The same holds if $h + g = e$ and $k + g = e$.
3. More generally, you can cancel in equations over groups: if $g + h = g + k$ then $h = k$. Similarly, $h + g = k + g$ implies $h = k$ too.

Exercise. ($\star\star\star$) *Rules in groups.* Prove Proposition 1.2.

Note: This is a three-star exercise and therefore strictly OPTIONAL. Being able to do axiomatic proofs is not part of the requirements for this course.

1.3 Associativity and commutativity

If you are given a sum like $1+2+3$, do you stop to wonder if it's meant to be $(1+2)+3$ or $1+(2+3)$? Probably not, you know that this "doesn't matter" — which is just another way of saying that normal addition is associative. However, $(1-2)-3$ is not the same as $1-(2-3)$. Compilers need to be told things like this. Most programming languages have a table of all their infix operators somewhere, each listed with its precedence and whether it is left- or right-associative. In the C language, $+$ and $-$ are left-associative so $a+b+c$ will be compiled as $(a+b)+c$ and $a-b-c$ is $(a-b)-c$ as you would expect. Since $+$ on integers is associative, an optimizing compiler is free to rearrange the sum. For the purpose of this course, the terms "left-associative" and "right-associative" don't exist — either an operation is associative in which case it makes no difference, or an operation is not associative and we will bracket it or not write it "infix" at all.

Something that we have omitted to mention until now is that $a + b = b + a$ on normal and fixed-bitlength integers. This is not the same thing as asssociativity — it is a new property called commutativity and we will see groups later on that are not commutative.

Definition 1.3. A group $(G, +)$ is commutative if for all a, b in G we have $a+b = b+a$.

A group that is commutative is sometimes also called Abelian after the mathematician N. H. Abel.

Exercise. ($\star$) *Basic groups.*

1. Why is $(\mathbb{N}, +)$ not a group, which group laws does it obey and which ones not? (Note: a structure that obeys the same subset of group laws as $(\mathbb{N}, +)$ is called a monoid.)
2. Why is $(\mathbb{Q}, \cdot)$ — the rational numbers with multiplication — not a group?
3. Why can you not have a group with 0 elements?
4. Can you have a group with one element?
5. Describe what a group with two elements must look like.

1.4 The groups $(\mathbb{Z}, +)$ and $(\mathbb{Z}_n, +)$

The first example of a group is $(\mathbb{Z}, +)$, the usual integers with addition. The group axioms are just some of the usual rules of arithmetic. Taking pairs, triples or any other vectors of fixed length of integers with component-wise addition (i.e. $[2, 3] + [4, 5] = [2 + 4, 3 + 5] = [6, 8]$ etc.) also gives a group. For vectors of integers of length n, this group is called $(\mathbb{Z}^n, +)$. The natural numbers $\mathbb{N}$ with addition are not a group, whether or not you include 0.

A slightly more interesting group is the group $(\mathbb{Z}_n, +)$ for any natural number n which has as its elements the numbers $\{0, 1, \ldots, n-1\}$. Addition works as follows: add two numbers normally; if the result is n or larger then you "wrap around": subtract n until you get a valid group element again. In the group $(\mathbb{Z}_8, +)$ for example, $2 + 2$ is still 4 but $6 + 6$ would normally give 12 so we subtract 8 and also get 4. This is a group (for any $n > 0$) but it takes a bit of number theory to show it. The neutral element is always 0 and the inverse of a number x in $(\mathbb{Z}_n, +)$ is the number $n - x$. The inverse of 7 in $(\mathbb{Z}_8, +)$ for example is $8 - 7 = 1$ since $1 + 7$ is 0 in this group.

An `unsigned int(eger)` datatype with n bits and its addition operation typically implement the group $(\mathbb{Z}_{2^n}, +)$.

WARNING. The $+$ in $(\mathbb{Z}_n, +)$ is a different operation to the $+$ in $(\mathbb{Z}, +)$! For $\mathbb{Z}_8$, the former operation has $6 + 6 = 4$ whereas the latter has $6 + 6 = 12$. If we wanted to be really precise, we could use a different symbol like $+_8$ for the operation in $\mathbb{Z}_8$ but we won't usually do this.

Exercise. ($\star\star$) *Basic properties of groups.* Over the integers $\mathbb{Z}$, complete the following table, placing a tick or a cross in each cell to indicate whether or not the operator has the property:

	$+$	$-$	$\times$	$\div$
associative				
commutative				
has neutral el.				
has inverses				
is group operation				

Exercise. ($\star\star$) *Addition tables.* For a finite group, we can describe the group operation by means of an addition table. Complete the following tables to make the operations into group operations. Note: there is exactly one way to do it in each case. You may use associativity to find this way but you do not have to check it everywhere!

$+$	♠	♣	♢	♡
♠				
♣	♣			
♢			♠	
♡				♠

$+$	A	B	C	D	E
A			B		
B					
C		C	E		D
D			A		
E					

- ($\star\star$) Which rules have you used to fill in the tables?
- ($\star\star\star$) How would you derive these rules from the definition of a group?

1.5 Order of a group

While we're working on groups we'll throw in the concept of an order:

Definition 1.4. The order of a group $\mathbb{G}$ is defined as follows: if the group's set G has a finite number n of elements, the order is n. If the group has infinitely many elements, the order is infinite (written ∞).

So $(\mathbb{Z}, +)$ has order ∞ and $(\mathbb{Z}_n, +)$ for $n > 0$ has order n.

Exercise. ($\star\star$) *Labelling with numbers.* For the two groups in the last exercise (addition tables), in one of them the elements can be labelled with numbers $0, 1, 2, \ldots$ such that group addition is exactly addition modulo the group order on these labels.

- Find a labelling for one of the two groups.
- Give a counter-argument to show why the other group cannot be labelled this way.

1.6 The "modulo n" operation

To better understand the group $(\mathbb{Z}_8, +)$ and how it relates to the better-known $(\mathbb{Z}, +)$, imagine all natural numbers written out in $n = 8$ columns (this works for any n of course):

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
...							

This table gives us a new way to compute in $\mathbb{Z}_8$: to add two numbers, add them normally, look up the result in the table then take the column heading as the result. For example sum above, $6 + 6 = 12$ and 12 is in column **4** so the result is 4. This lookup-column-name operation is important enough that we give it a name: to look up the column of a number x in an 8-column table like this is called "x modulo 8", written "$x \pmod 8$". Since going up one cell in the table subtracts 8, this table does the same job as the "subtract 8 until below 8" rule earlier.

This table is also an example of an equivalence relation, which we will discuss later.

1.7 Division with remainder

Neither the "repeatedly subtract 8" rule nor the table is really useful when we have to deal with large numbers. Instead, we invoke a fact of number theory.

Proposition 1.5 (Division with remainder). For any integer a and any positive natural number b there is exactly one pair (q, r) where q is any integer and r is an element of $\{0, 1, 2, \ldots, b-1\}$ making the equation $a = q \cdot b + r$ hold.

We call q the quotient and r the remainder of dividing a by b and write $q = a \text{ div } b$ and $r = a \bmod b$. In programming languages, these operations are often written $q = a \,/\, b$ and $r = a \,\%\, b$.

If we take a b-column table as above, quotient and remainder become row and column (starting both counts with 0 and including extra rows above for the negative integers). For example, in the 8-column table the number 12 can be found in row 1 and column 4:

	col 0	**col 1**	**col 2**	**col 3**	**col 4**	**col 5**	**col 6**	**col 7**
	...							
row -1	(-8)	(-7)	(-6)	(-5)	(-4)	(-3)	(-2)	(-1)
row 0	0	1	2	3	4	5	6	7
row 1	8	9	10	11	12	13	14	15
row 2	16	17	18	19	20	21	22	23
	...							

Which is another way of saying $12 \text{ div } 8 = 1$ and $12 \bmod 8 = 4$, giving us $12 = 1 \cdot 8 + 4$. The operation "12 mod 8" that we defined in proposition 1.5 does the same as $12 \pmod 8$ which we defined with the table.

We have defined three equivalent ways of computing modulo 8 (or modulo any other integer $n > 1$):

- Compute normally then add/subtract 8 repeatedly until you hit something between 0 and 7 (which is $8 - 1$).
- Compute normally then look up the result in an 8-column table and take the column number (starting to count with column 0, of course).
- Compute normally then do a division by 8 and take the remainder.

The last method is of course the "official" one. If an integer a leaves remainder 0 when dividing with remainder by b, we say that "a is a divisor of b", "a divides b" or "b is a multiple of a".

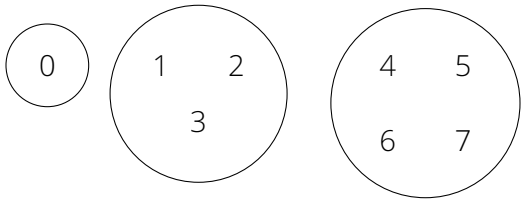
1.8 Equivalence relations and classes

As a next step working towards the formal definition of $(\mathbb{Z}_n, +)$ (which will come with a proof that it is actually a group), we introduce equivalence relations and classes.

A relation R on a set S can be thought of as a function that takes two elements a, b of S and outputs a boolean: `true` if a, b are "in relation R" and `false` if a, b are "not in relation R". There are, as always in Algebra, at least two different notations. It should be clear what $R(a, b) = \texttt{true}$ means; this is somtimes simply written $R(a, b)$ and pronounced "a, b are in relation R". The opposite is written $\neg R(a, b)$. Another way

of writing a relation is as an infix operator: $a\,R\,b$. This notation is common for relations like $=, \leq, <$: we usually write $a \leq b$ and not $\leq (a, b)$.

An equivalence relation is a relation that divides a set up into different subsets, called "classes". For example, here is one way to divide up the integers 0 to 7 into three classes:



The relation R that creates these classes returns `true` on any two numbers in the same class and `false` otherwise, e.g. $2\,R\,3$ holds but not $2\,R\,4$ (we could write this $2 \not R\, 4$, in the same way that we write $\neq$ to mean that the $=$ relation does not hold).

For a relation to split a set into classes, it must satisfy three conditions.

Definition 1.6 (equivalence relation). A relation R on a set S is an equivalence relation if it has these three properties.

reflexive For any element a of S we have $a\,R\,a$.

symmetric For any elements a, b of S if $a\,R\,b$ then also $b\,R\,a$.

transitive For any elements a, b, c of S if $a\,R\,b$ and $b\,R\,c$ then also $a\,R\,c$.

The smallest equivalence relation on any set S (that has the fewest number of inputs which produce the output `true`) is the equality relation $=$. The largest equivalence relation is the relation R that always returns `true`, so all elements of S end up in the same class.

If we have a set S and an equivalence relation $\sim$ (a symbol commonly used for equivalence relations), we can form a new set from the classes formed by $\sim$. This is called taking "S modulo $\sim$" and written $S/\sim$. We will soon see that taking an integer modulo another is a special case of taking a set modulo an equivalence relation. For the example relation in the diagram above, if we call the relation $\sim$ then we have

$$\{0, \ldots, 7\}/\sim \quad = \quad \{\{0\}, \{1, 2, 3\}, \{4, 5, 6, 7\}\}$$

so we have built a set with three elements, which are themselves sets. We could consider a function c that sends each element of S to its class, so $c(1) = c(2) = \{1, 2, 3\}$. There is a better way to work with classes however: choosing one element to represent each class.

1.9 Representative elements and $\mathbb{Z}_n$

Often it is helpful to pick names for the equivalence classes which are based on one of the elements that they contain. This is called choosing representative elements. Back to our 8-column table for an example:

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
...							
(-8)	(-7)	(-6)	(-5)	(-4)	(-3)	(-2)	(-1)
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
...							

We can interpret this table as an equivalence relation $\sim_8$ which divides the integers into 8 classes (the 8 columns), i.e. $a \sim_8 b$ if and only if a and b are in the same column. In this example, $c(1) = \{\ldots, (-7), 1, 9, 17, \ldots\}$. We can choose the "row zero" as representative elements, i.e. we give the 0-th column the label 0 and so on. We write the map that takes an element x to the representative element of its class as $[x]$.

Definition 1.7. For a fixed modulus n, the operation $[x]$ is the map $\mathbb{Z} \to \mathbb{Z}$ that sends x to its remainder modulo n. If necessary, we can also write $[x]_n$ to make the modulus clear.

◇ Formally, choosing representative elements means choosing a function Z from $\mathbb{Z}/\sim_8$ to $\mathbb{Z}$ that maps each class (which is a subset of $\mathbb{Z}$) to one of its elements. For example, $Z(c(1)) = Z(\{\ldots, (-7), 1, 9, 17, \ldots\}) = 1$. If it is clear which representatives are meant, we can omit making this step explicit and just write $\mathbb{Z}/\sim_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$. The [ ] function is just the composition $Z \circ c$.

1.10 Computing in $\mathbb{Z}_n$

To compute in $\mathbb{Z}_n$, the basic idea is "compute normally and reduce modulo n". We can optimize when we reduce: as long as we reduce at the end, we can reduce as little or as much as we want in between. For example, we can reduce whenever our intermediate results get too big for our liking.

Proposition 1.8. For any $a, b \in \mathbb{Z}$ we have

- $[[a]] = [a]$.
- $[a + b] = [[a] + b] = [[a] + [b]]$.
- $[[a + b] + c] = [(a + b) + c]$.

Since addition in $\mathbb{Z}$ is commutative, rules 2 and 3 also give $[a + b] = [a + [b]]$ and $[a + [b + c]] = [a + (b + c)]$. The important thing to remember in all rules is that you can never eliminate the outermost [ ] reduction. The proof of all of these rules is via the official definition of division with remainder. We discuss this in more detail in the next, optional section.

Exercise. *Division with remainder.*

1. ($\star$) Divide 256 by 15 with remainder.
2. ($\star$) Find an example of a modulus n and two numbers a, b such that $[a]+[b] \neq [a + b]$ where [ ] is taking the remainder modulo n.
3. ($\star\star$) For a fixed positive integer n, we know that every number $z \in \mathbb{Z}$ has exactly one pair (q, r) such that $r \in \{0, 1, \ldots, n-1\}$ and $z = q \cdot n + r$. Let's write $[[x]]$ for the operation that sends z to its pair (q, r).

   The pairs (q, r) of this form constitute a group with the "obvious" addition: call the addition in this group ++. Then for any numbers z, z' in $\mathbb{Z}$, if $(q, r) = [[z]]$ and $(q', r') = [[z']]$ then (q, r)++$(q', r') = [[z + z']]$.

   Describe the addition operation in this group in terms of addition in $\mathbb{Z}$ and reduction modulo n. What is the neutral element, what is the inverse of an element?

1.11 ◇ Formal construction of $\mathbb{Z}_n$

◇ Using the equivalence relation $\sim_n$ we can define $(\mathbb{Z}_n, +_n)$ formally:

Definition 1.9 ($\mathbb{Z}_n$). Let $n > 0$ be a positive integer. The equivalence relation $\sim_n$ is defined as $a \sim_n b$ if and only if a and b leave the same remainder when dividing by n.

The set $\mathbb{Z}_n$ is this set $\{0, 1, \ldots, n-1\}$ of representatives of the equivalence classes of $\mathbb{Z}/\sim_n$.

The group $(\mathbb{Z}_n, +_n)$ is the set $\mathbb{Z}_n$ with the following addition: $a +_n b := [a + b]$.

We have chosen to write $+_n$ to make clear that we are defining a new operation that differs from the usual $+$ on $\mathbb{Z}$ (in fact, it differs even on $\mathbb{Z}_n$). In the last definition, we claimed that $\sim_n$ was an equivalence relation on $\mathbb{Z}$ and that $(\mathbb{Z}_n, +_n)$ defined this way was a group. This section proves the above claims.

Proposition 1.10. The relation $\sim_n$ is an equivalence relation on $\mathbb{Z}$ for any integer $n > 0$.

Proof. We need to check the three properties. The relation $\sim_n$ is certainly reflexive because every number has the same remainder as itself when dividing by n. It is also symmetric because if r is the a and s is the remainder of b and $r = s$ then $s = r$ too (in other words, $=$ is symmetric). Transitive takes a tiny

bit more work. Let's take any integers a, b, c with $a \sim_n b$ and $b \sim_n c$. Then we can write a in exactly one way as $a = \alpha n + r$ for $r \in \{0, \ldots, n-1\}$ and b in exactly one way as $b = \beta n + s$ and c in exactly one way as $c = \gamma n + t$. Since $a \sim_n b$ we have $r = s$ and since $b \sim_n c$ we have $s = t$, from which we conclude $r = t$ (since $=$ is transitive) so $a \sim_n c$ too. QED.

Proposition 1.11. For an integer $n > 0$, $(\mathbb{Z}_n, +_n)$ is a group.

Proof. We have three conditions to check. For associativity, the equation to check is

$$a +_n (b +_n c) = (a +_n b) +_n c$$

We can write this as $[a + [b + c]] = [[a + b] + c]$. The idea is to eliminate the inner [ ] in both cases, then use associativity of $+$ on $\mathbb{Z}$, as we sketched in the rules in the last section.

To show $[x + y] = [x + [y]] = [[x] + [y]]$, write $x = \xi \cdot n + r$ and $y = \eta \cdot n + s$ in the unique way with $r, s \in \mathbb{Z}_n$. Then $[x + y]$ is the remainder of $(\xi + \eta) \cdot n + (r + s)$ modulo n which does not depend on ξ, η but is simply $[r + s]$. (Exercise: why do we need these brackets and can't simply write the remainder as $r + s$?). The term $[x + [y]]$ can be seen to be the remainder of $\xi \cdot n + r + s$ modulo n and again the ξ component does not influence the remainder so we get $[r + s]$, the same for $[[x] + [y]]$.

Back to our equation $[a + [b + c]] = [[a + b] + c]$: if r, s, t are the remainders of a, b, c modulo n then as a direct consequence of the above rule, this equation is equivalent to $[r + (s + t)] = [(r + s) + t]$ and the two sides are the same since $+$ on $\mathbb{Z}$ is associative.

The neutral element is the class $[0] = 0$: for any element a of $\mathbb{Z}_n$, we have $a +_n [0] = [a + [0]] = [a + 0] = [a]$ and $[0] +_n a = [[0] + a] = [0 + a] = [a]$, using that 0 was netural under $+$ in $\mathbb{Z}$.

The inverse of an element a in $\mathbb{Z}_n$ is $[n - a]$. We show $a +_n [n - a] = [0]$: this sum is $[a + n - a] = [n] = [0]$ and the same argument shows $[n - a] +_n a = [0]$. QED.