Mod/Div Turing Machine (Assessed Individual Coursework, Foundations 2 (F29FB), Spring 2022)

Joe Wells

2022-03-14 (Monday)

Contents

1 CHANGES TO THIS DOCUMENT SINCE ITS INITIAL RELEASE

Some fixes have been made to this assignment since it was initially released. You can find where these changes have been made by searching for the following words with all uppercase letters: UPDATED, FIXED, ADDED, CLARIFIED.

2 general information

UPDATED DUE DATE:

Dubai students: Due by 2022-03-16 T 23:59 +04:00.

Edinburgh students: Due by 2022-03-16 T 15:30 +00:00.

ADDED CLARIFICATION: The Edinburgh and Dubai students have different due times due to mandatory per-campus rules on due times.

This coursework is 18% of the mark for the course.

This is individual work. Failure to reference, collusion, and plagiarism are all FORBIDDEN. Read the university's guidelines to understand what these offenses are.

The standard university late submission guidelines are in force for this coursework.

You must submit via Canvas. Submission via email is FORBIDDEN.

3 the rmoddiv function

Some uses of **bold font** indicate references to terminology that is also discussed or defined in the lecture notes.

Let rmoddiv (Reversed **MODulo** operation and **integer DIVision**) be the unique **function** such that:

- rmoddiv $\in \mathbb{N}^2 \nrightarrow \mathbb{N}^2$
- (0, d) $\notin$ dom(rmoddiv) for every d $\in \mathbb{N}$ (i.e., rmoddiv(0, d) is "undefined")
- for every (s, d) $\in ((\mathbb{N} \setminus \{0\}) \times \mathbb{N})$, it holds that rmoddiv(s, d) = (d mod s, d div s) $\in \mathbb{N} \times \mathbb{N}$

NOTE: The variables s, d, r, q are used to represent the **diviSor**, the **diviDend**, the **Remainder**, and the **Quotient** of integer division.

4 main task: implement rmoddiv with a Turing machine

YOUR MAIN TASK IS: Design a **Turing machine** M_g such that rmoddiv = unaryIO$_{2,2}$(M_g), where the meaning of this equation is as follows:

When s, d $\in \mathbb{N}$ and your Turing machine M_g is started on a **tape** t that contains only **blank tape symbols** except for s consecutive 1's placed on the tape starting at position 0 (the initial **head position**) and going to the right, followed by a blank tape symbol, followed by d consecutive 1's, the outcome is as follows:

- If rmoddiv(s, d) = (r, q) $\in \mathbb{N} \times \mathbb{N}$, then the **computation sequence** must return the result by **halting** with a **final tape** t' that contains only blank tape symbols except for r consecutive 1's starting at the final head position and going to the right, followed by a blank tape symbol, followed by q consecutive 1's.
- If rmoddiv(s, d) is not defined, then your Turing machine (TM) must signal that the result is undefined by doing one of these two things:
  - The computation sequence may halt with a final tape t' that is an invalid output tape when expecting two natural numbers as output. There are three ways to do this:
    * Leave a **symbol** other than 1 or ^ on the tape.
    * Leave a 1 on the tape to the left of the final head position.
    * Leave more than two numbers on the tape, i.e., if the final head position is h, there is a blank at position b, a blank at position c, and a 1 at position e where $h \leq b < c < e$.
  - The computation sequence may **diverge** (not halt).

So for example, because rmoddiv(4, 15) = (3, 3), if your Turing machine M_g is started with the TM head at the leftmost 1 on the tape

```
1111^111111111111111
```

which is the number 4 followed by the number 15 in **unary** notation, your TM should finally halt with the TM head at the leftmost 1 on the tape

```
111^111
```

which is the number 3 followed by the number 3 in unary notation.

ADDED EXAMPLE: As another example, because rmoddiv(2, 9) = (1, 4), if your Turing machine M_g is started with the TM head at the leftmost 1 on the tape

```
11^111111111
```

which is the number 2 followed by the number 9 in unary notation, your TM should finally halt with the TM head at the leftmost 1 on the tape

```
1^1111
```

which is the number 1 followed by the number 4 in unary notation.

5 slight variations on the definition of Turing machine for this coursework

In this assignment, a TM **state** is any string that matches the regular expression "[A-Za-z][-A-Za-z0-9_<>]*". (This regular expression is valid in the main standards: POSIX extended, POSIX basic, and Perl.) This means that a state is a non-empty sequence of characters, begins with a Latin letter, and can have after the first character Latin letters, digits, - (the hyphen), _ (the underscore), and < and > (the less-than and greater-than symbols). If you want to follow the lecture notes, you can write, e.g., q_{12} as q_12.

For the purposes of connecting this coursework to the formalism in the lecture notes, we consider q_k to be the state whose name is of the form q_DIGITS where DIGITS is the standard decimal representation of $k \in \mathbb{N}$. For each state whose name does not match the pattern q_DIGITS, we will allocate an unused k and identify the state with q_k.

You must name your **initial state** q_0. If you submit a TM with no **actions** that **start** at state q_0, you will be deemed to have submitted a useless TM that does nothing.

NOTE: The use of q as part of a state identifier is unrelated to the use of q as a quotient.

In this assignment, a **symbol** must be one of these 37 characters:

```
^ 0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

We identify s_0 with ^ and s_1 with 1 and assume there is a similar identification for the other 35 symbols.

ADDED CLARIFICATION: In your `report.pdf` file, you may freely assume that ^ as a tape symbol is equal to s_0 and 1 as a tape symbol is equal to s_1. You may also freely assume there are corresponding equalities between the other tape symbols allowed for the assignment (e.g., 0, 2, 3, A, B, etc.) and the formal symbols of the lecture notes (e.g., s_2, s_3, s_4, s_5, s_6, etc.). You do not need to declare any of this.

NOTE: The use of s as part of a symbol identifier unrelated to the use of s as a divisor.

You must use ^ (the caret, a.k.a. the circumflex accent) as the blank tape symbol and 1 (the digit 1) as the symbol for the digit 1 in the unary input/output.

6 files to submit

ADDED FILE: You must submit exactly the following three files, all mandatory:

- `report.pdf`
- `turing-machine-action-table.txt`
- `authorship-declaration.docx` (ADDED FILE)

6.1 `report.pdf`

You must submit 1 file named exactly "`report.pdf`".

This file must be a PDF file.

Everything must be properly typeset. Handwriting is FORBIDDEN (except, when making nodes and edges of your graph, you may use software drawing tools that smooth your lines and make your circles exactly circular).

NEXT SENTENCE CLARIFIED:

The first line of this file must read exactly "Mod/Div Turing Machine Individual Coursework, F29FB, Spring 2022, H00123456", where H00123456 must be replaced by your student ID number.

Immediately after the first line must follow the following numbered items with these exact numbers ("1", "2", "3", "4", "5", and, optionally, "6"):

1. Item 1 must be your action table (i.e., your Turing machine) converted to graph format and presented beautifully. The initial state must be clearly marked.

   Item 1 must use at most 2 pages (preferably 1).

2. Item 2 must be exactly (no more, no less) a mathematical expression that represents your action table in correct mathematical notation.

   Item 2 must use at most half a page.

3. Item 3 must present the computation sequence of your TM on the input values 3 and 5 using the course's unary input/output conventions. The sequence must be annotated with a brief discussion of how your TM handles this input and the phases it goes through while doing this.

   Item 3 must use at most 2 pages (preferably 1).

4. Item 4 must be a clear, precise, and brief explanation in English of how your TM accomplishes, or fails to accomplish, the computation of the function rmoddiv. This discussion must include exactly these lettered sub-items with these exact letters ("A", "B", "C", "D", "E", "F"):

   A. Subitem A must briefly summarize in at most 5 sentences how your TM accomplishes the computation of rmoddiv.

   B. Subitem B must briefly discuss the symbols you use (including ^ and 1) and what they mean for your TM. If you use a symbol for multiple purposes, you must list all purposes.

   C. Subitem C must briefly discuss at most 3 (preferably fewer) key loops in your TM that involve more than one state. Discuss what each of these loops accomplishes and how this contributes to computing rmoddiv. It is a good idea to identify 1 or 2 key invariant properties that always hold at some specific loop points.

   D. Subitem D must briefly discuss all failures to compute rmoddiv for particular inputs. It is FORBIDDEN in this subitem to mention or rely in any way on the output of the checker program supplied by the lecturer. You must discuss all failures for any pair of inputs s and d where s and d both range from 0 through 50 (inclusive). You should have a good idea of the failure pattern of your TM for pairs of input numbers outside this range and you should also present this clearly. You must present the failure cases in at most 3 categories and explain why your TM fails for each category (e.g., if your TM fails on 200 pairs of numbers, we do not want 200 separate discussions but instead at most 3, preferably fewer).

      ADDED CLARIFICATION: If you write "I know it fails (or doesn't fail) because the checker program says so" then this portion of your report will be ignored and this will count as not having written anything for that part of the assignment. The checker is there to provide support to you and your markers but it does not replace the requirement for actual understanding.

      ADDED CLARIFICATION: If you indeed do not have any failures, you do not need to write anything for subitem D. Note that if the marker suspects a Turing machine fails on a particular input, the marker might add test cases to confirm the failure. Note also that item 4 subitem D is marked together with several other parts, i.e., it does not have a mark of its own. (See section "marks" for details.)

   E. Subitem E must briefly discuss 1 or 2 of your most significant design choices. You should consider discussing not only choices that helped but also choices that might have made your work harder.

   F. Subitem F must briefly discuss the efficiency of your TM. You should consider, in relation to the value/size of the input, both (1) the number of steps in the computation and (2) the size of the portion of the tape that is actually used at any/all points throughout the computation.

   Item 4 must use at most 1.5 pages.

5. Item 5 must be a clear, precise, and brief explanation of how to build a TM to do the same job as the main task, but for rounding integer division instead of floor integer division.

   Here is the background to the task. R-div (Rounding integer DIVision) is defined as follows:

   d R-div s = round(d / s) for all d, s $\in \mathbb{R}$

   R-mod is defined so that the following is true:

   (d R-div s) $\times$ s + (d R-mod s) = d for all d $\in \mathbb{R}$ and all s $\in \mathbb{R} \setminus \{0\}$

   FIXED: In the assignment as initially released to students, in the above equation, two occurrences of "s" were erroneously written as "v". This was fixed on 2022-03-04.

   R-rmoddiv is defined as follows:

R-rmoddiv(s, d) = (d R-mod s, d R-div s) for all d, s $\in \mathbb{N}$

FIXED: In the assignment as initially released to students, in the above equation, one occurrence of "s" was erroneously written as "d". This was fixed on 2022-03-14.

Observe that it is impossible to construct a TM that computes R-rmoddiv for natural number inputs using the rules of unaryIO. Specifically, if s, d $\in \mathbb{N}$ and R-rmoddiv(s, d) = (r, q), then q $\in \mathbb{N}$ always holds but it can be the case that r $\in \mathbb{Z} \setminus \mathbb{N}$ (i.e., r can be negative which unaryIO does not support).

Now here is the actual task. The work on this item must be presented in a list of exactly these lettered sub-items with these exact letters ("a" and "b"):

(a) You must specify in at most 3 lines an alternate function R-xmoddiv which provides the same information as R-rmoddiv in its output but in a way that a TM can be built to compute it using the rules of unaryIO.
ADDED CLARIFICATION: It must be possible that R-xmoddiv = $\text{unaryIO}_{2,k}(M)$ for some k $\in \mathbb{Z}^+$ and some Turing machine M. This implies that the inputs and outputs of R-xmoddiv must be tuples of natural numbers, and that the inputs of R-xmoddiv must be the same as the inputs of R-rmoddiv.
ADDED CLARIFICATION: The output of R-xmoddiv(s,d) must allow a user of R-xmoddiv to do whatever they would be able to do with the output of R-rmoddiv(s,d) with as little additional effort as possible. The tuple of natural numbers returned by R-xmoddiv(s,d) must represent (as closely as you can) the result of R-rmoddiv(s,d).

(b) You must briefly describe in at most 4 sentences what alterations would be needed to your Turing machine M_g to create a new Turing machine M_h that would compute R-xmoddiv.

Item 5 must use at most half a page (and preferably much less).

6. Item 6 must contain anything you think is important to write as part of your submission but which is not covered by the above list items. This item is optional, but if present must be numbered 6 and must contain all material not in items numbered 1 through 5. There is no specific mark associated with this item, but if it turns out that this item is needed (e.g., you realize you need a bibliographic reference), leaving it out might reduce one of the mark components.

There must be nothing after the list of items specified above.

The `report.pdf` file must be at most 5 pages (preferably fewer). Any pages after the 5th page of `report.pdf` will be automatically removed before marking and will not be read by the marker. Do not include a title page.

6.2 `turing-machine-action-table.txt`

You must submit 1 file named exactly "`turing-machine-action-table.txt`".

6.2.1 required format and content of `turing-machine-action-table.txt`

Here is an example of the contents of a valid `turing-machine-action-table.txt` file:

```
# H00123456
#
# This TM erases the first unary number on the tape and puts a 1 in the
# space that was after that number.
#
# This TM computes the function f ∈ ℕ ↛ ℕ such that
# f(x) = 1 for all x ∈ ℕ.
#
```

```
# This TM also computes the function g ∈ ℕ² ↠ ℕ such that
# g(x, y) = 1 + y for all x, y ∈ ℕ.
#
{
  ((q_0, 1), (q_0, ^, R)),
  ((q_0, ^), (q_1, 1, 0)),
}
```

The `turing-machine-action-table.txt` file must contain the exact same TM that you present in `report.pdf`. There is no separate mark for the `turing-machine-action-table.txt` file.

Your TM will be tested for correctness of its input/output behavior for at least all pairs s and d of natural numbers from 0 to 50 (inclusive) and possibly a selection of pairs of numbers outside this range. Your TM will be allowed up to 0.2 seconds (using the lecturer's TM simulator) and 10 000 000 steps per computation sequence.

The file `turing-machine-action-table.txt` must have the following very precise format. A line can be empty, in which case it is ignored. A line can begin with "`#`", in which case it is a comment for any human reader and is not part of the TM. Comments in this file are primarily for the student author and the marker is unlikely to read them. The first non-empty non-comment line must contain exactly "`{`". A line can be of this format:

```
((state,symbol),(state,symbol,move)),
```

In this case, the first state and symbol are the current (old) state and symbol of an action, and the second state and symbol are the new state and symbol of the same action. The last non-empty non-comment line must contain exactly "`}`". There are no other formats for lines.

The first line must be a comment that contains your student ID number (which looks like `H00123456`) and nothing else.

All space (U+0020) and tab (U+0009) characters are ignored.

If you get the format of this file wrong, your TM will be automatically considered to fail all tests.

If the contents of this file fail to denote a function, your TM will be automatically considered to fail all tests.

If the file contains no actions for state `q_0`, then the automated testing will think your TM always halts immediately. You must name your initial state `q_0` (where the underscore "_" is significant).

If you fail to submit this file or get the format wrong, your mark will suffer because the automated testing will say that your TM handles every input incorrectly.

6.2.2 program for checking your `turing-machine-action-table.txt` file (STRONGLY RECOMMENDED)

There is a file you can download at this URL:

`https://www.macs.hw.ac.uk/~jbw/teaching/hw/F29FB/2021,-22/f29fb-2021,-22-cw-checker.py`

It is also available at and runnable from this file system location on the NFS file systems in Edinburgh of the school of Mathematical and Computer Sciences (MACS):

```
/home/jbw/f29fb-2021,-22-course-support/f29fb-2021,-22-cw-checker.py
```

NEXT SENTENCE CLARIFIED:

This is a Python program which you can use to check your `turing-machine-action-table.txt` file for syntactic and semantic correctness. You do not need to know Python to use it, but you do need Python installed on whatever computer you use it on. You can use the lecturer's installed copy by logging in using SSH or x2go from home to `ssh.macs.hw.ac.uk`, because Python is installed on that computer.

IT IS STRONGLY RECOMMENDED THAT YOU USE THIS PROGRAM MANY TIMES BEFORE YOU SUBMIT!!!

ADDED CLARIFICATION: Take the time to fix at least some of the errors you will inevitably discover! If you do not use the checker program, it is quite likely that you will have made a syntax error and your testing score will be zero and this will lower your mark substantially.

For further instructions, run the program and/or read the documentation near the beginning of the program.

6.3 ADDED FILE: `authorship-declaration.pdf`

You must submit 1 file named exactly "`authorship-declaration.pdf`".

This file must be a PDF file. Your authorship declaration MUST NOT be a `.docx` (OOXML) file!

You must create this file by the following process.

1. First, you must read and understand the material found at these two URLs:

   `https://www.hw.ac.uk/uk/students/studies/examinations/plagiarism.htm`

   `https://heriotwatt.sharepoint.com/sites/skillshub/SitePages/Academic-Integrity-and-Plagiarism.aspx`

2. Then, you must download the file at this URL:

   `https://www.macs.hw.ac.uk/~jbw/teaching/hw/F29FB/2021,-22/authorship-declaration.docx`

3. Then you must edit this file and fill in the blanks with the following information:

Course code and name:	F29FB Foundations 2
Type of assessment:	Individual
Coursework Title:	Mod/Div Turing Machine
Student Name:	your name
Student ID Number:	your ID number (which looks like "H00123456")
Student Signature	your signature (made by typing, not handwriting)
Date	the date you fill the form

   You must only fill in your signature if you agree with what the declaration in the file states. You must ensure that the declaration in the file with your signature added is true.

4. Then, after editing the `.docx` template to enter your details, you must save the result as as PDF file named exactly "`authorship-declaration.pdf`".

You must ignore the instructions in the file that tell you to "insert it into your coursework file in front of your title page". You must NOT include the contents of this file in your `report.pdf` file. You must instead submit it as a separate file.

7 marks

Let Rn mean item n in the file `report.pdf`. Let R4X mean subitem X of item 4 in the file `report.pdf`. Let TMAT mean the file `turing-machine-action-table.txt`.

Let TM mean R1, R2, and TMAT taken together. TM is your Turing machine.

Let Understanding mean R3, R4A, R4B, R4C, and R4D, possibly augmented by useful annotations in R1, which present your Understanding of how/why your TM computes or fails to compute rmoddiv.

The marking will be as follows:

8	accuracy of computation of rmoddiv by your TM
7	correctness, clarity, insight, and completeness of your presentation of your Understanding
3	clarity and all other aspects of correctness (e.g., syntax, notation) of your TM
3	R4E (wisdom you may have gained from or further developed in the coursework)
2	R4F (understanding of efficiency)
2	R5 (extrapolation of your learning to a related but different problem)
2	following all instructions
27	total marks possible (will be multiplied by 2/3 to make up 18% of your course mark)

8 questions and answers about the coursework

8.1 what is div?

Recall the definition of integer division from Lecture Notes 2. "div" is integer floor division. d div s = $\lfloor$d / s$\rfloor$, i.e., d div s is the **floor** of d / s.

8.2 what is floor?

Recall the definition of floor from Lecture Notes 2. Formally, $\lfloor \cdot \rfloor$ is a function such that for all x $\in \mathbb{R}$ it holds that x $\geq \lfloor$x$\rfloor$ and $\lfloor$x$\rfloor \in \mathbb{Z}$ and for all k $\in \mathbb{Z}$ it holds that if x $\geq$ k then x $\geq \lfloor$x$\rfloor \geq$ k. Informally, the floor of x is the nearest integer to x that is not greater than x.

8.3 what is mod?

Recall the definition of the modulo operation from Lecture Notes 2. "mod" is the modulo operation. "mod" is defined by the relationship that holds with div, namely (d div s) $\times$ s + (d mod s) = d.

8.4 what is round?

Recall the definition of round from Lecture Notes 2. Formally, round is a function such that for all x $\in \mathbb{R}$ it holds that |x - round(x)| $\leq$ 0.5 and round(x) $\in \mathbb{Z}$ and if |x - round(x)| = then round(x) is even.

8.5 why is the function named "rmoddiv"?

The combination of integer division and the modulo operation is named `divmod` in some programming languages. The "r" is for "Reversed" because rmoddiv takes its arguments in the opposite order to how div, mod, and divmod work. The "moddiv" instead of "divmod" is because it returns the mod result first and the div result second which is opposite of "divmod".

8.6 why is rmoddiv(0, d) undefined?

rmoddiv(0, d) is undefined for any s $\in \mathbb{N}$ because both integer division and the modulo operation are undefined when the divisor is 0. "d div 0" is undefined because "d / 0" is undefined. "d / 0" is undefined because if d $\neq$ 0,

there is no number q such that $0 \times q = d$, and because if $d = 0$, then there are q and q' such that $q \neq q'$ and $0 \times q = d$ and $0 \times q' = d$ (in fact there are an infinite number of distinct solutions). "d mod 0" is undefined because mod is defined together with div.

8.7 ADDED QUESTION: I think rmoddiv(2, 9) = (1, 4), but I am uncertain. how can I be sure?

Let's work this through:

- $2 \in \mathbb{N}$. (Easy.)
- $2 \neq 0$. (Also easy.)
- $2 \notin \{0\}$. (By definition of singleton sets.}
- Therefore, $2 \in \mathbb{N} \setminus \{0\}$. (By definition of set subtraction.)
- $9 \in \mathbb{N}$. (Easy.)
- Therefore $(2, 9) \in (\mathbb{N} \setminus \{0\}) \times \mathbb{N}$. (By definition of set product.)
- Therefore, because we know the condition is true in the 3rd rule of rmoddiv's definition, we also know this must be true: rmoddiv(2, 9) = (9 mod 2, 9 div 2).
- 9 div 2 = $\lfloor 9 / 2 \rfloor = \lfloor 4.5 \rfloor = 4$. (By definition of div and floor.)
- The rule that defines mod is: (d div s) $\times$ s + (d mod s) = d.
- If we solve for (d mod s) in this rule, we get this more direct definition of mod: (d mod s) = d - (d div s) $\times$ s.
- Thus, 9 mod 2 = 9 - (9 div 2) $\times$ 2 = 9 - 4 $\times$ 2 = 9 - 8 = 1. (By arithmetic from facts we have just confirmed.)
- Thus, rmoddiv(2, 9) = (1, 4). (By combining facts we have just confirmed.)

Therefore, you can be confident that rmoddiv(2, 9) = (1, 4).

8.8 ADDED QUESTION: I am still confused about what the correct results of calling the function rmoddiv are. Help!

There is a Python implementation of rmoddiv in the checker program. Also, if you feed the checker a bad TM and use the `--comprehensive-test` option, you can get it to produce a list of all results for pairs of natural numbers up to 50. That should help you figure out what the correct results are.

8.9 what is unaryIO?

The unaryIO operator is defined in Lecture Notes 9 and builds on the definitions in Lecture Notes 8. The unaryIO operator is merely the precise mathematical statement of the conventions for unary input/output used for TMs from Lecture Notes 6 (which introduces TMs) onwards.

Here is a short summary of what rmoddiv = $\text{unaryIO}_{2,2}(M_g)$ implies:

- If s, d $\in \mathbb{N}$ and rmoddiv(s, d) = (r, q) $\in \mathbb{N}$ and your Turing machine M_g is started on a tape t such that t = unaryInput(s, d) (i.e., s consecutive 1's are placed on the tape starting at position 0 and going to the right, then a blank, then d consecutive 1's), then the computation sequence must halt with a final tape t′ and a final head position i such that renumberTape(t′, i) = unaryInput(r, q) (i.e., the final tape must contain only blank tape symbols except for r consecutive 1's starting at the final head position and going to the right, followed by one blank, followed by q consecutive 1's).
- If s, d $\in \mathbb{N}$ and rmoddiv(s, d) is not defined and your Turing machine M_g is started on a tape t such that t = unaryInput(s, d), then your TM must signal that the result is undefined by doing one of these two things:
  - The computation may halt with a final tape t′ such that t' $\neq$ unaryInput(r, q) for every r, q $\in \mathbb{N}$. An easy way to do this is to leave a symbol other than 1 or ^ on the tape. Other ways are leaving 1's to the left of the final head position and leaving more than two numbers on the tape (i.e., leaving a ^ symbol at the final head position or somewhere to the right of it, and to the right of that somewhere another ^ symbol, and to the right of that somewhere a 1).
  - The computation may diverge (not halt).

8.10 how big is your TM solution?

The lecturer's solution uses 12 active states (and 1 inactive state), 30 actions, and 4 symbols (including ^ and 1).

NOTE: There is no expectation that students will design a TM that small (although no doubt one or two very clever students will find smaller solutions). This information is intended to be helpful, e.g., if your TM has more than 100 states, this will help you understand that it might be a bit more complicated than it needs to be and that you might benefit from simplifying.

8.11 are 10 000 000 steps enough to compute rmoddiv(50, 50)?

The lecturer's solution calculates rmoddiv(100, 100) in less than 21 000 steps, and calculates rmoddiv(s, d) for all natural numbers s and d from 0 to 50 (inclusive) in less than 6 000 steps. So limiting your TM to 10 000 000 steps is reasonable. Your TM is allowed to be more than 1600 times less efficient than the lecturer's.

8.12 my computation sequence for `report.pdf` item 3 won't fit on 2 pages!

Have you tried multiple columns? You can probably fit 50 lines per page, and probably 3 columns, which allows for showing 300 configurations. If needed, you can use a smaller font for this item.

For comparison, the lecturer's implementation runs for 85 steps on this input and thus encounters 86 configurations, none of which need to show more than 11 tape positions.

ADDED CLARIFICATION: If your computation sequence diverges you must justify in writing how you know this (without referring to any use of the checker program the lecturer supplies) and you must write as much of the sequence as you can fit in two pages.

ADDED CLARIFICATION: If your computation sequence halts but is extremely long, you must include an initial portion of the sequence and the number of steps. If it can fit in less than a page, you must also include the final configuration.

8.13 ADDED QUESTION: what is the smallest font size I may use and what is the smallest margin width I may use?

The smallest font size you may use is 10 points, with standard line spacing corresponding to a 10 point font size. The smallest margin width you may use is 2.5 cm.

If you are having difficulty fitting your report within the limit using these parameters, perhaps you should try to see if you can say the same things with fewer words.

8.14 can I use symbols other than ^ and 1?

You may use any of the 37 symbols allowed in this coursework during the middle of your computation but if your TM halts with a final tape that contains anything other than two natural numbers starting at the final position of the head, the output will not count as two natural numbers and will instead count as the function being undefined for that input.

8.15 what should my TM do on an initial input tape that has symbols other than ^ or 1 on it, or more than two numbers on it, or 1's to the left of the starting position?

By the rules of unary$IO_{2,2}$, it does not matter what your TM does on input tapes that do not encode two natural numbers in unary notation. For example, in the past, some students have had their TM's print a message on the tape when this happens. The assignment does not care.

8.16 can I include my name or ID or username or the course code or the date in my file names?

Variation in capitalization, spelling, or punctuation of any file name is FORBIDDEN. You must not include your name, computer account, email address, or student ID number in the name of any file you submit. The only allowed file names are:

- `report.pdf`
- `turing-machine-action-table.txt`
- `authorship-declaration.docx` (ADDED FILE)

8.17 can I submit a `.docx` or `.odt` or `.gif` or `.png` or `.svg` file?

Files in the `.docx` or `.odt` or `.gif` or `.png` or `.svg` formats are FORBIDDEN.

The only allowed formats are plain text (`.txt`) and PDF (`.pdf`).

8.18 what file formats are allowed?

`report.pdf` The file `report.pdf` must be a PDF file.

`turing-machine-action-table.txt` The file `turing-machine-action-table.txt` must be a plain text file in the UTF-8 character encoding.

UTF-16 and UTF-32 (all variants) are FORBIDDEN and if your `turing-machine-action-table.txt` uses one of these encodings, it will be automatically considered to fail all tests.

NOTE: ASCII (a.k.a. US-ASCII) is a subset of UTF-8, so if some program reports your file is ASCII that means it is also UTF-8.

ADDED FILE: `authorship-declaration.pdf` The file `authorship-declaration.pdf` must be a PDF file.

Your authorship declaration MUST NOT be a `.docx` (OOXML) file! You MUST save the `.docx` template as a PDF file after you enter your details!

8.19 can I use different item numbers or subitem letters or bullets in `report.pdf`?

No.

Item 1 must be numbered "1". Item 2 must be numbered "2". Item 1 must come before item 2. In other words, `report.pdf` must be nothing but a big numbered list from 1 to 5 (or possibly 6) with item 4 containing a lettered sublist from A to F.

8.20 how can I draw the graph in `report.pdf`?

There must be node (state) and edge labels.

You may (and it is a good idea) place explanatory labels on the graph identifying groups of nodes that act together in a certain way.

The graph of your TM must be produced with some kind of graph-drawing software and must have properly typeset node and edge labels.

Drawing by hand is forbidden, except that it is okay if you draw by hand provided you use software that turns your drawing into perfect circles/ovals/etc. and straight lines and smooth curves.

Your lecturer sometimes draws TM graphs using `dot` (easier) and sometimes uses LaTeX with PSTricks (harder, but can make it look nicer). There are a lot of other options also.

It is okay to include an image file generated by your graph-drawing software.

8.21 what state names can I use?

Your start state must be named q_0. It is a good idea if you pick descriptive state names for your other states.

The states in your `turing-machine-action-table.txt` file must match exactly the states in your graph (i.e., the node labels there) and other parts of your submission, with the exception that, if you wish, for states with names of the form `q_0`, `q_1`, `q_2`, etc., in your `turing-machine-action-table.txt` file you may use node labels of the form q_0, q_1, q_2, etc., in your graph and other parts of your submission. In other words, you may represent the final part of a state that looks like "_x" by a subscript like $_x$.

8.22 what is a mathematical expression?

`report.pdf` item 2 is asking for an *expression*, not a statement or statements.

Here are examples of expressions:

- 1
- 7 + 9
- $\{2, 3\} \cup \{3, 4\}$
- f(3)
- $\{(1, 2)\}$ (ADDED EXAMPLE)

Expressions stand for mathematical objects/values.

Here are examples of statements (which are formally called "formulas") which are NOT expressions:

- x = 1

- f(x) = x + 1 if $x \in \mathbb{N}$
- x = y or f(x) = 3

Statements assert/claim mathematical truths.

ADDED INFORMATION: You might also find this information helpful:

- There is an example of a specific Turing machine as an expression in Lecture Notes 11.
- There are examples of expressions that evaluate to different Turing machines under different circumstances in Lecture Notes 12.
- There are example of expressions for specific functions in Lecture Notes 2.
- There are examples of expressions that evaluate to different functions under different circumstances in Lecture Notes 5.

8.23 ADDED QUESTION: what is a loop that involves more than one state?

First, we define some terminology needed for the answer to this question. A Turing machine *action* ((q_i, s_k), (q, s_l, X)) *starts* at q_i and *ends* at q_j. Actions A_1 and A_2 (which might be the same action) are *consecutive* exactly when A_1 ends at the same state that A_2 starts at. A *loop* is a non-empty sequence of actions A_1, A_2, ..., A_k such that (1) A_i and A_{i+1} are consecutive for every $i \in \mathbb{N}$ such that $1 \leq i < k$, and (2) A_k and A_1 are consecutive.

Now we give the answer to the question: A loop involves more than one state exactly when at least one of the actions in it does not start and end at the same state.

8.24 what goes in item 6 in `report.pdf`?

It's there to prevent things that shouldn't go in items 1 through 5 from going in those items. For example, if you feel helplessly compelled to include a poem in every coursework submission, item 6 is where it belongs. If you find you need one or more bibliographic references, they go in item 6.

8.25 what line ending characters can my `turing-machine-action-table.txt` file use?

Lines may be ended by LF (U+000A) or by CR (U+000D). A CR LF sequence counts as ending a line and then an empty line. Because empty lines are ignored in `turing-machine-action-table.txt`, it does not cause any problems to interpret CR LF this way.

8.26 what are the precise syntax rules for the `turing-machine-action-table.txt` file?

All space (U+0020) and tab (U+0009) characters are ignored.

A BNF-like grammar for the file format (assuming all spaces and tabs are already removed) is:

```
tm ::= ignored* begin line* end ignored*
ignored ::= newline | comment newline
newline ::= "[\n\r]"
comment ::= "#[^\n\r]*"
begin ::= "\{" newline
line ::= ignored | action
```

```
action ::= "\(\(" state "," symbol "\),"
            "\(" state "," symbol "," move "\)\)," newline
state ::= "[A-Za-z][-A-Za-z0-9_<>]"
symbol ::= "[A-Z0-9^]"
move ::= "[LR0]"
end ::= "\}" newline
```

8.27 what tape position does the TM head start at?

In every TM computation sequence, the head always starts at position 0 on the tape.

8.28 if the numbers 3 and 5 have been written on the tape, what tape positions do they occupy?

When using the unary input/output conventions of this course (which are specified as the unaryIO operator), if two natural numbers 3 and 5 are written on the tape, then there are eight digit one symbols on the tape at positions 0, 1, and 2 for the first natural number and at positions 4, 5, 6, 7, and 8 for the second natural number and all other tape positions including position 3 contain a blank symbol.

8.29 ADDED QUESTION: what tape corresponds to the 2-tuple of natural numbers (0, 1)?

The tape that is unaryInput(0, 1) looks like this where the leftmost depicted "^" is at position 0 and all non-depicted tape cells contain the blank symbol "^":

```
^1
```

Let t be this tape. It holds that $\text{unaryOutput}_2(t) = (0, 1)$.

8.30 if my TM passes the tests you provide do I need to write anything for item 4 subitem D?

You do not need to describe failures if there are none.

If there is even one pair of natural numbers s and d such that your TM computes rmoddiv(s, d) wrong, you must report this, regardless of whether the checker program supplied by your lecturer detects this.

8.31 is it good enough if my TM passes the tests you provide?

The assignment is to implement rmoddiv correctly, and to do this with reasonable efficiency for input numbers up to 50. The assignment is NOT just to pass a specific testing regimen.

The checker program that you have been supplied is only an aid. You MUST NOT rely on your TM only being checked in the way that program checks it.

8.32 "`f29fb-2021,-22-cw-checker.py --comprehensive-test`" takes 10 minutes. is this too long?

The 0.2 seconds time limit is per computation sequence. The command

```
f29fb-2021,-22-cw-checker.py --comprehensive-test
```

runs 2601 computation sequences (as of the version of 2022-02-05). So 10 minutes is averaging less than 0.2 seconds per computation sequence. That is fine. The purpose of the time limit is to allow us not to have to wait a week to get the testing results for all the students before we start marking.

8.33 when running `f29fb-2021,-22-cw-checker.py` how can I get the tape positions to line up vertically?

You might want to run the checker program like this:

```
f29fb-2021,-22-cw-checker.py --run-on-tape '^^^^^^@111^11111^^^^^'
```

The extra blanks on either side in the initial tape prevent the checker from reallocating memory for the tape in the middle of the computation, which will make your display look nicer because each tape position will always have the same horizontal position. Of course you should choose the number of blanks on each side of the initial input based on how much space your TM actually uses. The @ specifies the location that will become position 0 on the tape.

8.34 ADDED QUESTION: how do I run the checker program on my own computer?

Here are some simple instructions. You don't have to do it exactly like this, but this is likely to work.

1. Make sure Python 3 is installed on your computer. Linux computers generally already have Python 3. Many software packages for Windows are implemented in Python and come with a Python interpreter and library, so Python might already be installed on a Windows computer. If you install the Windows Subsystem for Linux (WSL) it will almost certainly include Python. If you don't already have Python on your computer, you can install it by following the instructions at:

   `https://wiki.python.org/moin/BeginnersGuide/Download`

2. Let PYTHON stand for either "python3" (Linux or macOS) or "py" (Windows) depending on which operating system you are using.

3. Make a directory/folder for your Turing machine testing. Let DIR stand for this directory.

4. Download the checker program `f29fb-2021,-22-cw-checker.py` and put a copy in directory DIR.

5. Put your `turing-machine-action-table.txt` file in directory DIR.

6. Start a terminal/console window with a shell (command-line interpreter) running in it.

7. Change the current directory of the shell in the terminal to DIR. If you are using Linux, the command to do this is (where DIR must be replaced by the directory's actual name):

   ```
   cd D
   ```

8. If you want to test your Turing machine on a pair (NUM1, NUM2) of input natural numbers, run this command (where PYTHON, NUM1, and NUM2 must be replaced by the correct strings):

   ```
   PYTHON f29fb-2021-22-cw-checker.py --run-on-numbers NUM1 NUM2
   ```

9. If you want to test your Turing machine like it will be tested after you submit it, run this command (where PYTHON must be replaced by the correct string):

   ```
   PYTHON f29fb-2021-22-cw-checker.py --comprehensive-test
   ```

8.35 ADDED QUESTION: how do I run the checker program on `ssh.macs.hw.ac.uk`?

Here are some simple instructions. You don't have to do it exactly like this, but this is likely to work. These instructions assume your computer is running Linux, so you will have to adapt them if you are running macOS or Windows. On Windows, you might benefit from installing the Windows Subsystem for Linux (WSL).

1. Start 2 terminal/console windows with a shell (command-line interpreter) running in them on your own computer. Let T1 be the first terminal window and T2 be the second.
2. In T1, log in to `ssh.macs.hw.ac.uk` with a command like this:

   ```
   ssh ssh.macs.hw.ac.uk
   ```

3. In T1, make a directory for your Turing machine testing with a command like this:

   ```
   mkdir f29fb-turing-machine/
   ```

4. In T2, copy your `turing-machine-action-table.txt` file to `ssh.macs.hw.ac.uk` with a command like this:

   ```
   scp turing-machine-action-table.txt ssh.macs.hw.ac.uk:
   ```

5. In T1, move the your `turing-machine-action-table.txt` file into the directory you made with a command like this:

   ```
   mv turing-machine-action-table.txt f29fb-turing-machine/
   ```

6. In T1, change the current directory of the shell running on `ssh.macs.hw.ac.uk` to the directory you made with a command like this:

   ```
   cd f29fb-turing-machine/
   ```

7. Let CHECKER stand for this file name:

   ```
   /home/jbw/f29fb-2021,-22-course-support/f29fb-2021,-22-cw-checker.py
   ```

8. If you want to test your Turing machine on a pair (NUM1, NUM2) of input natural numbers, run this command (where CHECKER, NUM1, and NUM2 must be replaced by the correct strings):

   ```
   CHECKER --run-on-numbers NUM1 NUM2
   ```

9. If you want to test your Turing machine like it will be tested after you submit it, run this command (where CHECKER must be replaced by the correct string):

   ```
   CHECKER --comprehensive-test
   ```

8.36 ADDED QUESTION: I still can't get the checker program to work! Help!

Please read the problem reporting guidelines at this URL:

`https://www.macs.hw.ac.uk/~jbw/#debug`

Following these guidelines, send a detailed problem report to Joe (the lecturer who wrote the checker program).

8.37 how should I write the configurations in the computation sequence in `report.pdf` item 3?

If you explain your notation for TM configurations properly, you may use any reasonable notation. The output format of the checker program is an example of a reasonable notation. It is a good idea to use a notation that fits everything in a configuration on one line, because that will take less space and be easier to read.

You may indicate the sequencing of the configurations (from first to last or from first to the point where you indicate it goes on forever) using any reasonable notation, provided you explain your notation.

You must not rely solely on color for indicating some part of the information, because your work might be marked by someone with some kind of color-blindness.

ADDED CLARIFICATION: It must be clear for each configuration what all of the components of the configuration are. You are responsible for understanding the definition of configurations in the course notes and knowing what a configuration consists of.

ADDED CLARIFICATION: The boundaries between distinct configurations on the same page must be clear. There should not be any configuration that is split across 2 pages.

8.38 should I make citations and bibliographic references?

Standard academic requirements apply to whether or not you must cite the work of others. Every student must strictly abide by Heriot-Watt's rules forbidding plagiarism and failure to reference.

If you include verbatim chunks of material written by others in any file you submit, you must clearly mark the beginning and end of each such chunk. For English language text the standard way to do this is with quotation marks. Furthermore, there must be a citation to the source next to the chunk (with page number or other appropriate means of finding the material within the source), and the source should be detailed in a bibliographic reference in `report.pdf` item 6.

8.39 ADDED QUESTION: what will satisfy the requirement for a PDF file?

In practice, if it obeys some version of the PDF standard and also can be displayed by the `evince` program version 40 or earlier, it's probably fine. In principle, please use only features defined in some version of the PDF specification excluding features defined in vendor-specific extensions (including extensions by Adobe) and also excluding features deprecated in any version of the specification. Versions of PDF through 1.7 are specified by Adobe Systems. Version 1.7 is also international standard ISO 32000-1:2008. Version 2.0 is specified only by the ISO and is ISO 32000-2:2020.

8.40 ADDED QUESTION: what is MACS?

"MACS" is an acronym that stands for "Mathematical and Computer Sciences". You are taking a course in the Computer Science Department which is a part of MACS. Some computers that are relevant for this assignment are run by MACS. The relevant computers are `www.macs.hw.ac.uk`, which is the web server where you get the assignment, and `ssh.macs.hw.ac.uk`, which is a Linux computer you can log in to via SSH and on which you can run a copy of the checker program if you don't want to run it on your own computer.

8.41 ADDED QUESTION: is a chart/graph useful for item 4 subitem F on efficiency?

Measuring the behavior of your program is better than having no knowledge of its time and space behavior. A chart/graph is a good way to present any measurements you make.

However, even better is knowing why and being able to explain this succinctly. The important thing is to demonstrate your understanding of the time and space behavior of your TM and why it has that behavior.

You might benefit from refreshing what you know from earlier courses about analysis of algorithms and algorithmic complexity.