COMP2034 – C++ PROGRAMMING

Mohammad Aazam

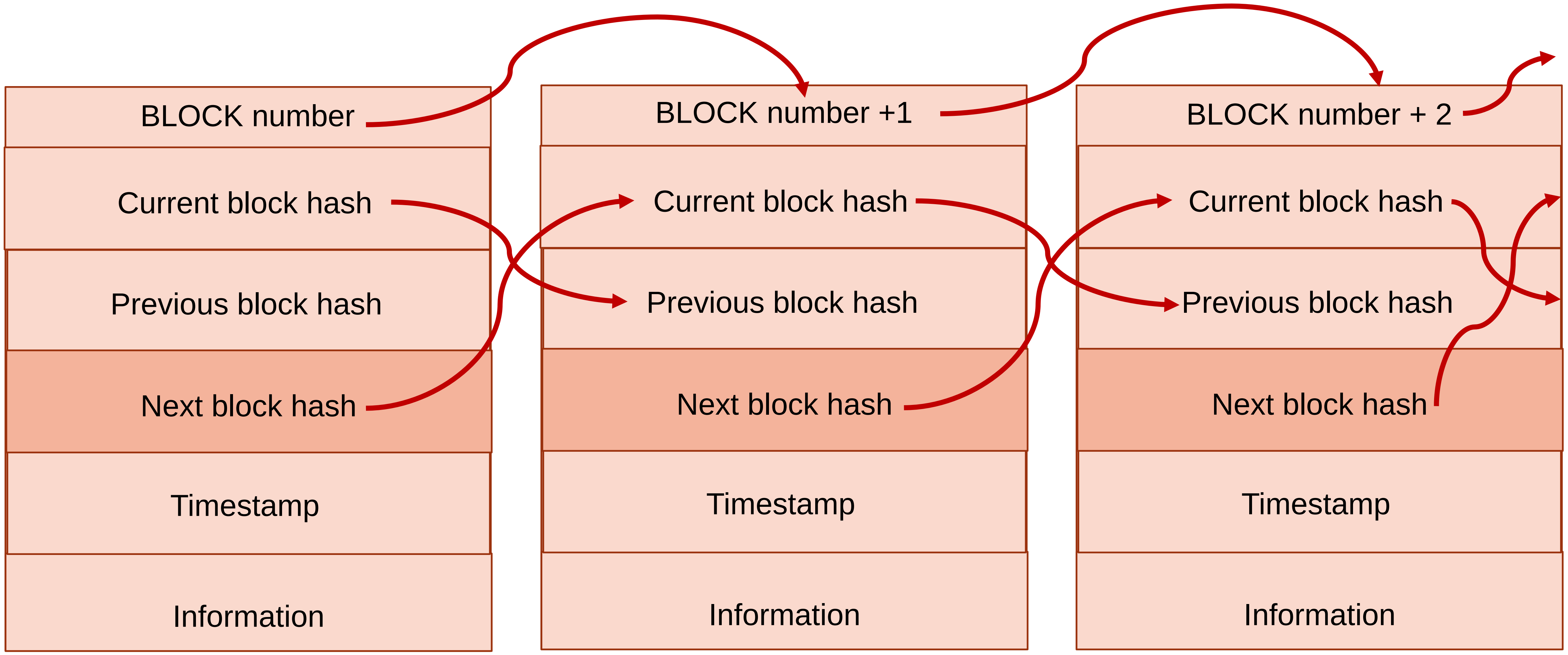
University of Nottingham (Malaysia)

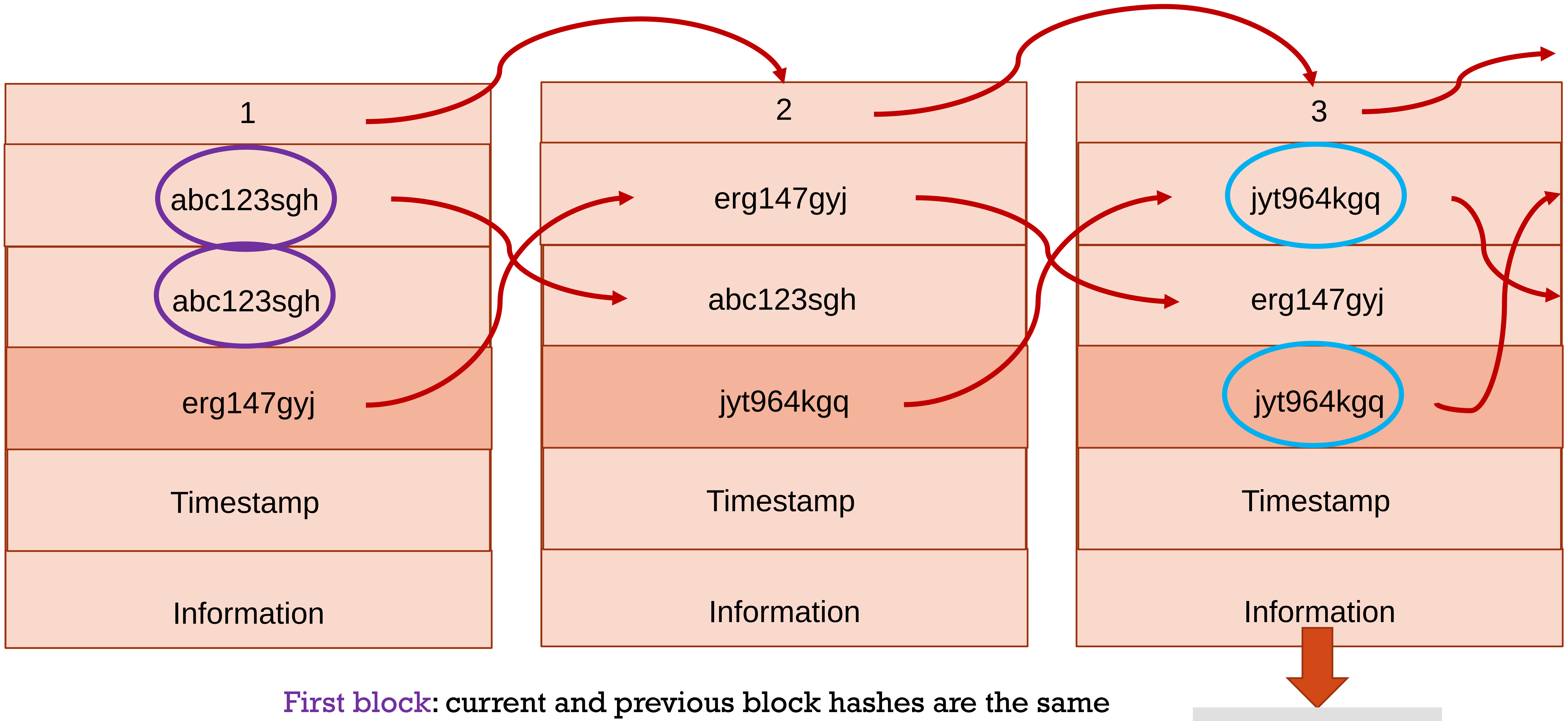
CW2: EXTENDING CW1

- In CW1, the core requirements were
  - To show connectivity of blocks in the blockchain
    - Previous block hash was the key in this regard
  - If a block was removed, disconnection was displayed
- In CW2, lets make our blockchain more dynamic

CW2: DYNAMIC BLOCKCHAIN

- A block should be connected to its previous as well as next block (something like doubly linked list)
- The blockchain should have multiple features
  - Such as, add, delete, edit/modify
  - Upon modification, the block should be rehashed and the entire blockchain should be updated (which means, all blocks should get updated of the new hash of the modified block)
- Include exception handling (what if values entered are incorrect or in a wrong format?) – I know we haven't studied exception handling yet, but you can use your creativity as much as you can

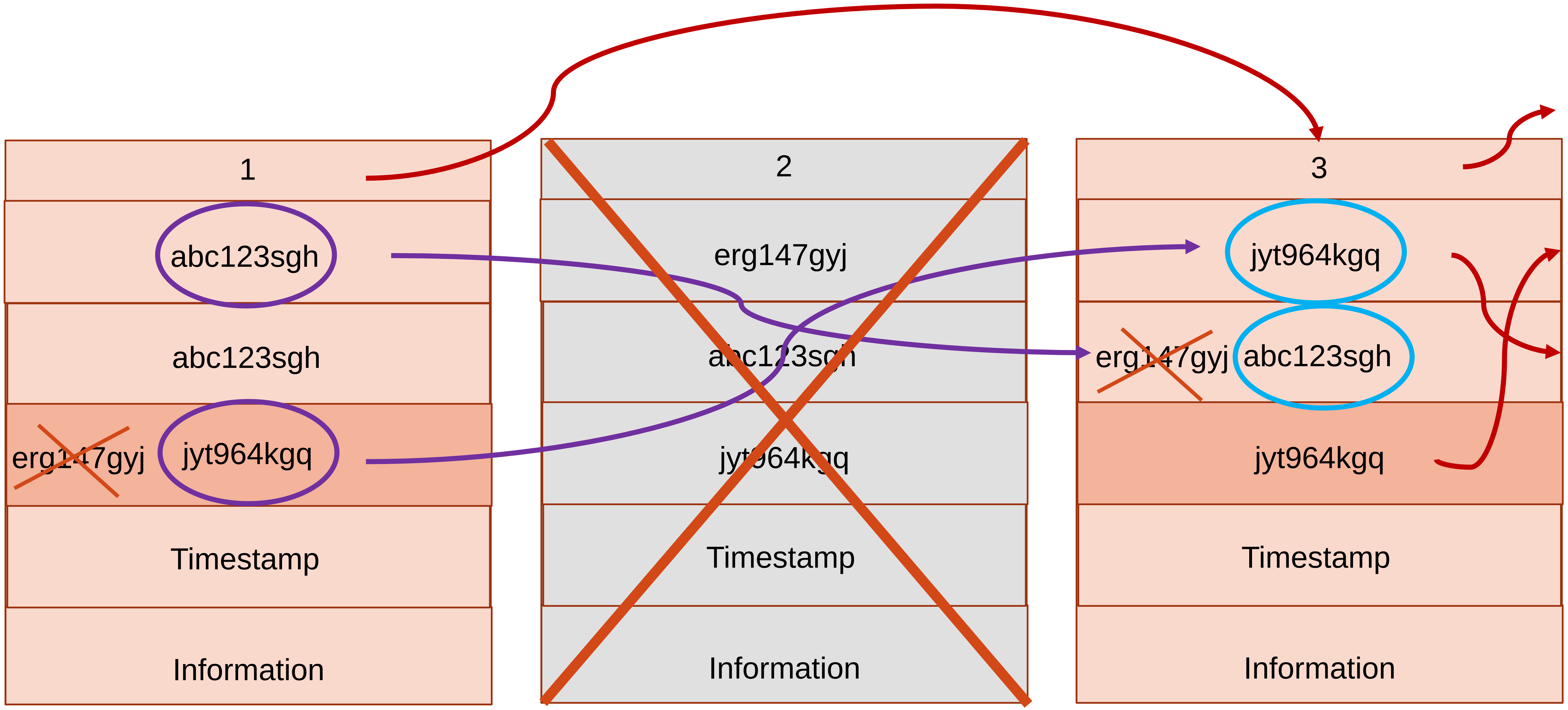

BLOCK number
Current block hash
Previous block hash
Next block hash
Timestamp
Information
BLOCK number +1
Current block hash
Previous block hash
Next block hash
Timestamp
Information
BLOCK number + 2
Current block hash
Previous block hash
Next block hash
Timestamp
Information



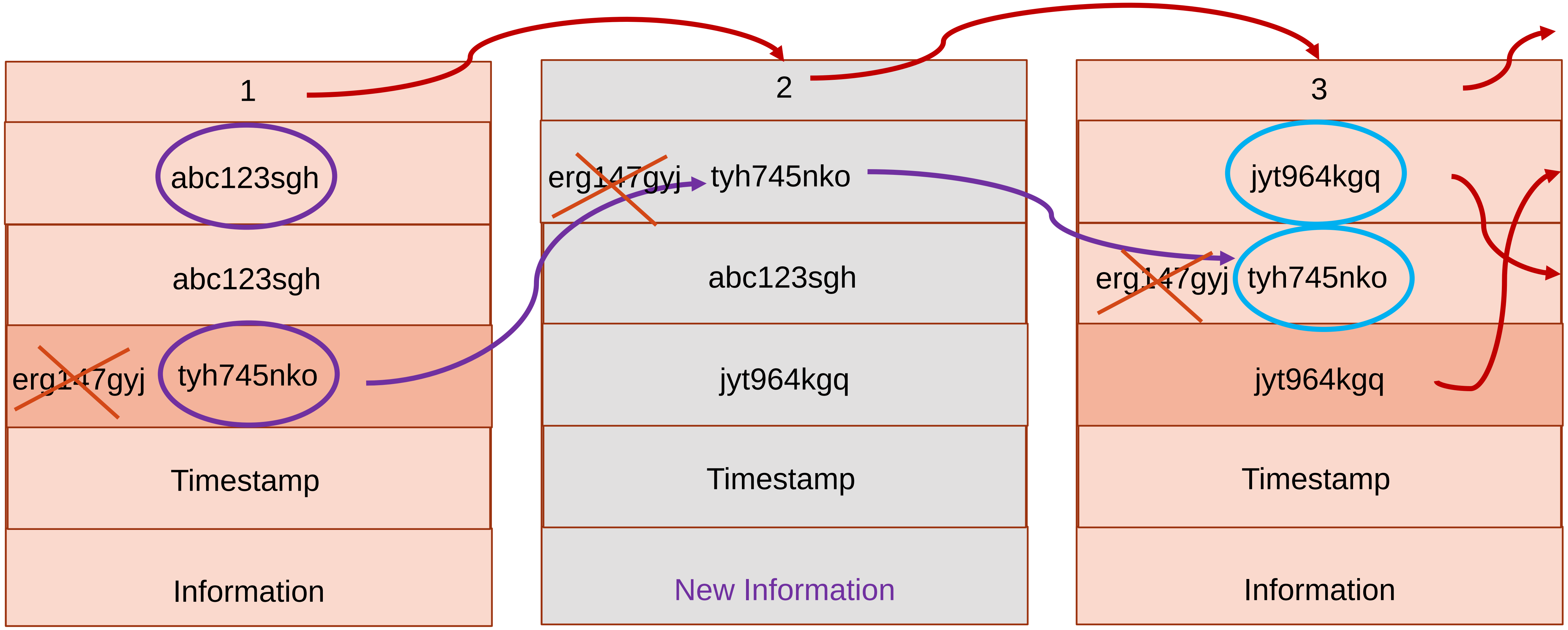
First block: current and previous block hashes are the same

Last block: current and next block hashes are the same

Your edu record (as in CW1)


1
abc123sgh
abc123sgh
~~erg147gyj~~
jyt964kgq
Timestamp
Information
2
erg147gyj
abc123sgh
jyt964kgq
Timestamp
Information
3
jyt964kgq
~~erg147gyj~~
abc123sgh
jyt964kgq
Timestamp
Information

DELETION



MODIFICATION

CW2: MANDATORY FEATURES

- 1. two-way block connection
  - As in doubly linked list (connection with previous and next)
- 2. dynamicity
  - Add, delete, modify, display
- 3. storage
  - Save the entered data
    - Whatever data is once entered should be saved in the program (you can save it on a separate file as well, such as in a .txt file. We haven't yet studied file manipulation in C++, but just in case you want to explore it. Otherwise, save inside of your program in anyway you like (e.g., array))
    - The saved data/block can be removed through the delete feature of your program
    - When the program is closed, it should retain any newly added blocks or any modifications in the blocks and display the updated blockchain every next time

CW2: OUTPUT

- When the program is run, it must display the following:
  - Pre-existing blockchain (this is your CW1 blockchain that contains your education's historical record, starting from grade 10 to the latest. There will be around 5 or so blocks in your so-far blockchain)
- Ask user to select from a menu options
  - Menu options must be as below (feel free to further show your creativity and user-friendliness):
  - Add, delete, modify, display
- If the user selects add, ask for the information required to create the new block (just as in CW1)
- Once new block is created, add it to the existing blockchain and inform the user (what's next is up to you. Make it user-friendly)
- If the user selects to modify blockchain, as for the block they want to modify
  - When the user selects a particular block, get the information of the block again, calculate new hash for it, replace it at the same spot in the blockchain, and update the entire blockchain (now the neighboring blocks' next/previous block hashes will be updated)
- If the user selects to delete a block, ask them which block they want to delete. Based on the selection, delete that particular block, show the result, rejoin the blockchain, and show the updated blockchain
- Always save any modification (e.g., update, delete, add)
- Display option should show the current blockchain

REMEMBER: addition is always at the end, as a new block.
While deletion and modification can be on any block within the existing blockchain.

CW2: SUBMISSION

- Submission is due on 05 April 2021 (23:59 midnight)
- Submission is via Moodle (link will be up soon)
- Submit the following:
  - Fully functional source file (.cpp) – if needed, include a readme file (for instructions regarding how to run your file(s))
  - Object file (.o)
  - Screenshots for all significant output that you want me to see and grade you on
  - Short report (same as CW1, the three points: overall experience, difficulties faced, new learning. No length limit (feel free to explain your journey but don't make it very lengthy)
- Each and every file you submit should follow the below naming convention
  - StudentID_StudentFullName_FileType
    - 202103_MohammadAazam_CW2.cpp
    - 202103_MohammadAazam_CW2.o
    - 202103_MohammadAazam_Report.pdf
    - Naming of screenshots is up to you (maybe you want to name them in a self-explanatory way)

CW2: GRADING

- The grading will be based one the following:
  - Adherence to the mandatory requirements
  - Cleanliness of the code
  - Proper commenting in the code (try to explain how you do something specific. Make it easy for someone new to modify your code)
  - Creativity (I know this is hard to explain and judge, but this is what makes it easier to incentivize a deserving student better)
  - User-friendliness (and how organized your output and overall application is)
  - Report (remember that report has only three sections and there's no length limit. Just try to make the best use of it by being creative, walk me through your experience, yet not being too detailed). Do not spend too much time on the report. It has the least weightage in the grading

DO NOT WORRY

- Do not feel burdened
- Enjoy the journey of learning (I hope it will be a good memory for you as well)
- Explore more and get yourself exposed to new things
- Take the challenge and overcome it
- Do not worry about grades, focus on your learning and enhancing your skills
- You can do it
- Yes, you can
- Do take care of your health
- Do get in touch with me if you need to discuss anything
- All the best