Part A:

1.

    1.store value “x: “ in register r1
    2.print the value of the r1 register that means print “x: “
    3.get user input and stores in the r2 register
    4.store value “y: “ in register r1
    5.print the value of the r1 register that means print “y: “
    6.get user input and stores in the r3 register
    7.if value of the r1 register equals value of r2 register then jump to line 11 else go to next line
    8.store value “x is not equal to y” in register r1
    9.print the value of the r1 register that means print “x is not equal to y “
    10.end the process
    11.store value “x is equal to y” in register r1
    12.print the value of the r1 register that means print “x is equal to y “

To print the “x: “ message in the stdout first it should load into a register. Line 1-2 is doing doing that work

After we get the input from the user it should store in a not used register.in line 3 we store the input value in a r2 register.

For other input we do the same thing mentioned above. Line 4-6 is for get the second input

Line 7 is for comparing the input values if they equals jump in to line 11 and load “ inputs are equal” msg into a register r1 and in line 8 print the value in the register r1.

If inputs are not equals then line 8-9 print the msg “x and y are not equals”. And line 10 end the process

So this code is correctly implemented.

2.

```
1.SET r1 “x:  “
2.PRINT r1
3.INPUT r2
4.SET r1 “y: “
5.PRINT r1
6.INTPUT r3
7.JUMPEQ r2 r3 15
```

```
8.JUMPTL r2 r3 12
9.SET r1 “x is greater than y”
10.PRINT r1
11.EXIT
12.SET r1 “x is less than y”
13.PRINT r1
14.EXIT
15.SET r1 “x is equal to y”
16.PRINT r1
```

3.

```
1.SET r1 “x:  “
2.PRINT r1
3.INPUT r2
4.set r3 0
5.set r4 1
6.set r1 “cough”
7.print r1
8.add r3 r3 r4
9.jumplt r2 r3 6
10.exit
```

4. By line ”main:”
   In this assembly language functions call by function name followed by a colon
   In this compare.s file their is a line “main:”, and it means calling the main function

5. Compiler store the two msg memory and after comparing inputs particular msg will load to a register and will be printed.there is an instruction (.cmpl) which is comparing the input values and if x>y then (.jpe) will jump to the else part.

Part B

1. Blockchain is a distributed ledger, which simply means that a ledger is spread across the network among all peers in the network, and each peer holds a copy of the complete ledger.

   Blockchain is the underlying technology that runs Bitcoin. It’s a distributed public ledger that records and saves a record of every single transaction. Blockchain technology allows payments and other digital information to be exchanged without the need to authenticate transactions through a central authority

2. A cryptographic hash function is an algorithm that can be run on data such as an individual file or a password to produce a value called a checksum.The main use of a cryptographic hash function is to verify the authenticity of a piece of data. Two files can be assumed to be identical only if the checksums generated from each file, using the same cryptographic hash function, are identical.

3. Cryptographic hash functions should be impossible to crack with current technologies. But this function can crack easily so this function cannot be used as a cryptographic hash function.

4.
   ```
   struct block{

       char previous_hash[256];

       char trans_indentifier[50];

        char trans_sender[50];
   ```

```
        char trans_recipient[50];

        int amount;

        char digital_signature_sender[100];

        char proof_of_work[100];

    };
```

Part C

1. 128 characters

2. Since the restriction of the Unicode code-space to 21-bit values in 2003, UTF-8 is defined to encode code points in one to four bytes, depending on the number of significant bits in the numerical value of the code point. The following table shows the structure of the encoding. The `x` characters are replaced by the bits of the code point. If the number of significant bits is no more than seven, the first line applies; if no more than 11 bits, the second line applies, and so on.

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	`0xxxxxxx`			
2	11	U+0080	U+07FF	`110xxxxx`	`10xxxxxx`		
3	16	U+0800	U+FFFF	`1110xxxx`	`10xxxxxx`	`10xxxxxx`	
4	21	U+10000	U+10FFFF	`11110xxx`	`10xxxxxx`	`10xxxxxx`	`10xxxxxx`

3. A continuation byte in UTF-8 is any byte where the top two bits are `10`.

They are the subsequent bytes in multi-byte sequences. The basic rules are this:

1. If a byte starts with a `0` bit, it's a single byte value less than 128.
2. If it starts with `11`, it's the first byte of a multi-byte sequence and the number of `1` bits at the start indicates how many bytes there are in total (`110xxxxx` has two bytes, `1110xxxx` has three and `11110xxx` has four).
3. If it starts with `10`, it's a continuation byte.

This distinction allows quite handy processing such as being able to back up from *any* byte in a sequence to find the first byte of that code point. Just search backwards until you find one not beginning with the `10` bits.

Part D

1. The starting character is "y" and after "e" and "s" and ending character is "s" and it is repeat one or more times.

2. The starting character is y and the e and s after whitespace and that whitespace will repeat one or more times.

3.
```
import re

    words = input("Say something!\n")

    p = re.compile("my name is (^([\w\-]+))", re.IGNORECASE)

    matches = p.search(words)

    if matches:

            print(f"Hey, {matches[1]}.")
```

```
else:

    print("Hey, you.")
```

4. EMMA

5. EMMA, EMMAEMMA, EMMAEMMAEMMA

6. EMMAA

Part E

1.
   ```
   INSERT INTO followers
   VALUES (
   SELECT id FROM users WHERE username=”max”,
   SELECT id FROM users WHERE username=”ileana”
   );
   ```

2.
   ```
   SELECT users.username
   FROM users
   LEFT JOIN followers ON users.id = followers.followee_id
   WHERE followers.folower_id IN
           (SELECT id FROM users WHERE username=”reese”)

   INTERSECT

   SELECT users.username
   FROM users
   LEFT JOIN followers ON users.id = followers.followee_id
   WHERE followers.folowee_id IN
           (SELECT id FROM users WHERE username=”reese”);
   ```

3.
   ```
   SELECT users.username
   FROM users
   LEFT JOIN followers ON users.id = followers.followee_id
   WHERE followers.folower_id IN
           (SELECT followee_id FROM followers WHERE IN (
                   SELECT id FROM users WHERE username=”reese”)
                   );
   ```

4. CREATE TABLE friendships(
```
sender_id INTEGER,
receiver_id INTEGER,
is_accepted INTEGER DEFAULT 0,
FOREIGN KEY (sender_id) REFERENCES users(id),
FOREIGN KEY (receiver_id) REFERENCES users(id)
);
```

5. INSERT INTO friendships(sender_id ,receiver_id)
```
VALUES(
        SELECT id FROM users WHERE username=”max”,
        SELECT id FROM users WHERE username=”ileana”,
);
```

6. UPDATE friendships
```
SET is_accepted = 1
WHERE IN sender_id (SELECT id FROM users WHERE username=”max”)
AND receiver_id(SELECT id FROM users WHERE username=”ileana”);
```

7. DELETE FROM users
```
WHERE username=”reese”;

DELETE FROM followers
WHERE IN follower_id(SELECT id FROM users WHERE username=”reese”)
OR followee_id(SELECT id FROM users WHERE username=”reese”);


DELETE FROM friendshps
WHERE IN receiver_id(SELECT id FROM users WHERE username=”reese”)
OR sender_id(SELECT id FROM users WHERE username=”reese”);
```

Part F

1.T3A4C2G1A3

2.sequences without repeating characters adjacently.
Ex: original version: ACGT (4 characters)
Compressed version A1C1G1T1 (8 characters)

3. Germany's flag will be compressed more. Because it's the whole column is a one-color sequence, therefore, fewer characters will be used. On the other hand, the Romania flag has three colors per row so it will require many more characters to represent a row.

4.11101011110

5.1110101110

6. Problem:

We can't separately identify two characters in this encoding method.

New Encoding method:

A-0
C-10
G-110
T-1110

Part G

1. **Sorted**

   Sort function running time depend on what sort algorithm am I going to use.

   If i use like bubble sort ,Insertion sort algorithms it will take **O(n^2)** . OrI will use

   sort like merge sort or quick sort it will get **O(nlg(n^2)).**

   Therefore we can get average running time as :**O(nlg(n^2)).**

   issorted:**O(n)**

2. In this case we must go through all elements in the list check either this list is sorted
   or not .There is no possible way to go through the list in constant time. It must depend on the size of the list.Therefore **issorted** can't do in constant time.

3. In the worst case it can be infinite running time.It can be running infinite.

4. In the best case first iterate can find sorted list
   Therefore lower bound is given sort function is :**O(n)**

5. In the worst case sorted list can find after checking all combinations.
   Therefore upper bound is given sort function is:**O(n.2^n)**