In [1]:
```
%run Imports.ipynb
```

```
Using TensorFlow backend.
```

Introduction

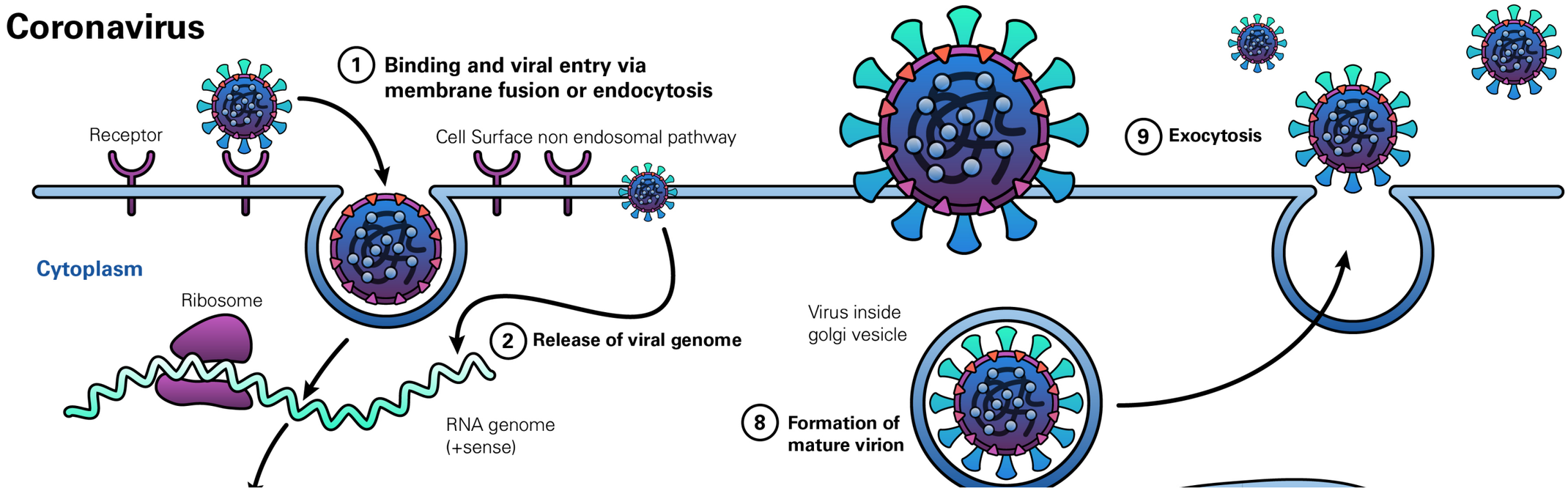
In this report, the biology of sars-cov-2 is discussed and the sars-cov-2 genome is analyzed. Deep learning will be used to predict the following nucleotide in the genome sequence.

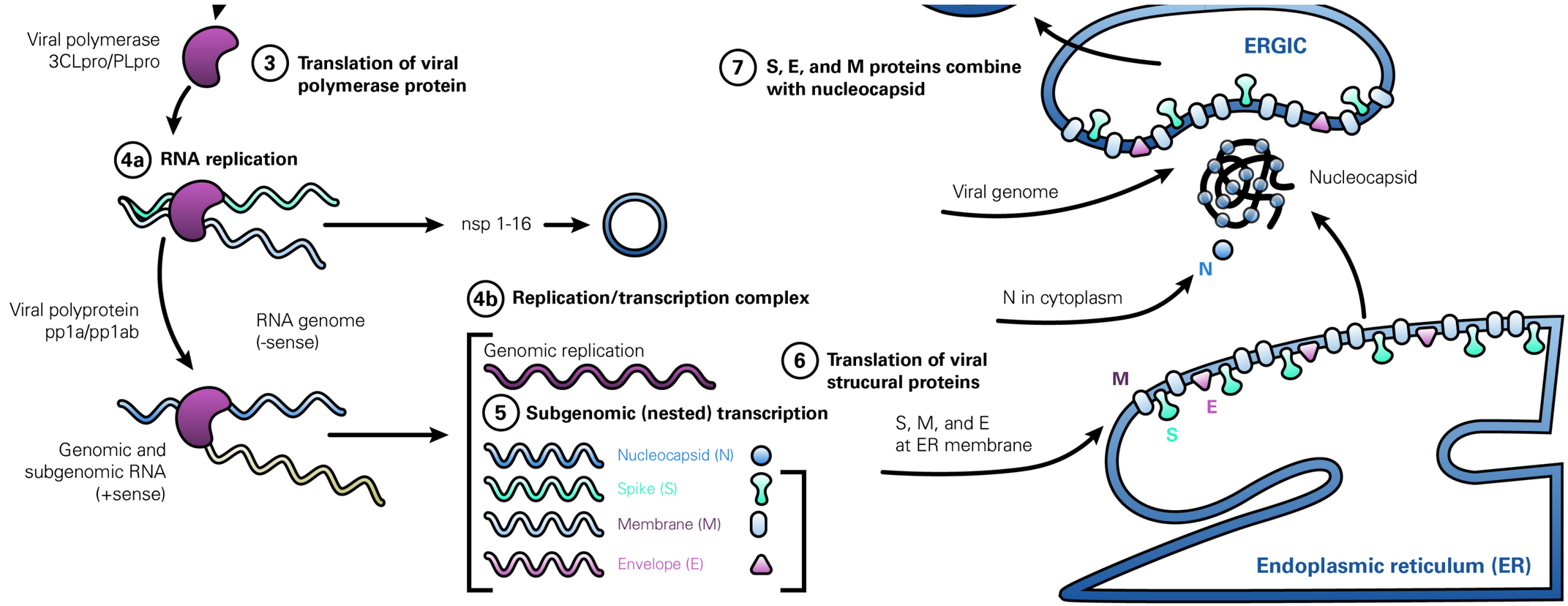
SARS-CoV-2 Biology

Coronaviruses are a sphere-like 20-sided polygon, icosahedron, shape and belong to the Nidovirales order. SARS-CoV-2 travels through the respitory tract and attaches itself to alveoli, air sacs, in the lungs. Coronaviruses are named after the dyed red spike glycoproteins (S) that cover them and attach these protiens by being activated by the host cell's proteases, ACE2 and TMPRSS2 ennzymes, at a sequential proteolytic cleavage S1 and S2 site to fuse with a host cell's enzymes to enter, endocytosis, the cell's plasma membrane to the cytosol. S1 is the end binding region of the S protein and S2 is the beam of the S protien. The spike glcoproteins can also attach to a cell's endosomes being activated by the host cell's CTSL enzyme. Inside of the icosahedron positive-sensed lipid bilayer envelope is a helical capsid that encloses an RNA genome (gRNA), which encodes the viral genetic information -- non-segmented-single-stranded RNA -- that is used during the lytic cycle to generate RNA polymerase, protease, and other proteins. The envelope is removed allowing genomic RNA to enter the cytoplasm where some of it is translated into pp1a and pp1ab proteins. Translation is when codons, groups of three nucleotides, move along a ribosome through tRNA that translates the codons to anticodons that generate an amino acid polypeptide chain of proteins. Both proteins are then cleaved by protease making 16 nonstructural proteins, NSPs. Some of the NSPs form a replication and transcription complex (RTC) in which gRNA is replicated and RNA-dependent RNA polymerase (RdRp) uses gRNA as a template to transcribe (-) subgenomic RNA (sgRNA) and then transcribe it back to (+) sgRNA, which is used as the genome of the new virus. sgRNA is translated into strutural proteins, wich are spike protein (S), envelope protein (E), membrane protein (M), and nucleocapsid protein (N). Nucleocapsid proteins assign the viral replication-transcription complexes (RTCs), where gRNA and sgRNAs are synthesized. Spike, envelope, and membrane proteins enter the endoplasmic reticulum, and the nucleocapsid protein encloses the sgRNA becoming a nucleoprotein complex. The proteins and nucleoprotein complex combine into a virus endoplasmic reticulum-Golgi apparatus, and then are excreted into a vesicle. The dyed yellow envelope proteins (E) and dyed orange membrane glycoproteins (M) assist in budding -- using the host cell's plasma membrane for envelope formation -- and then exiting, exocytosis, the host cell through lysis. The following is a diagram of the SARS-CoV-2 lifecycle,

In [4]:
```
Image(filename='Coronavirus_Life_Cycle.jpg')
```

Out[4]:

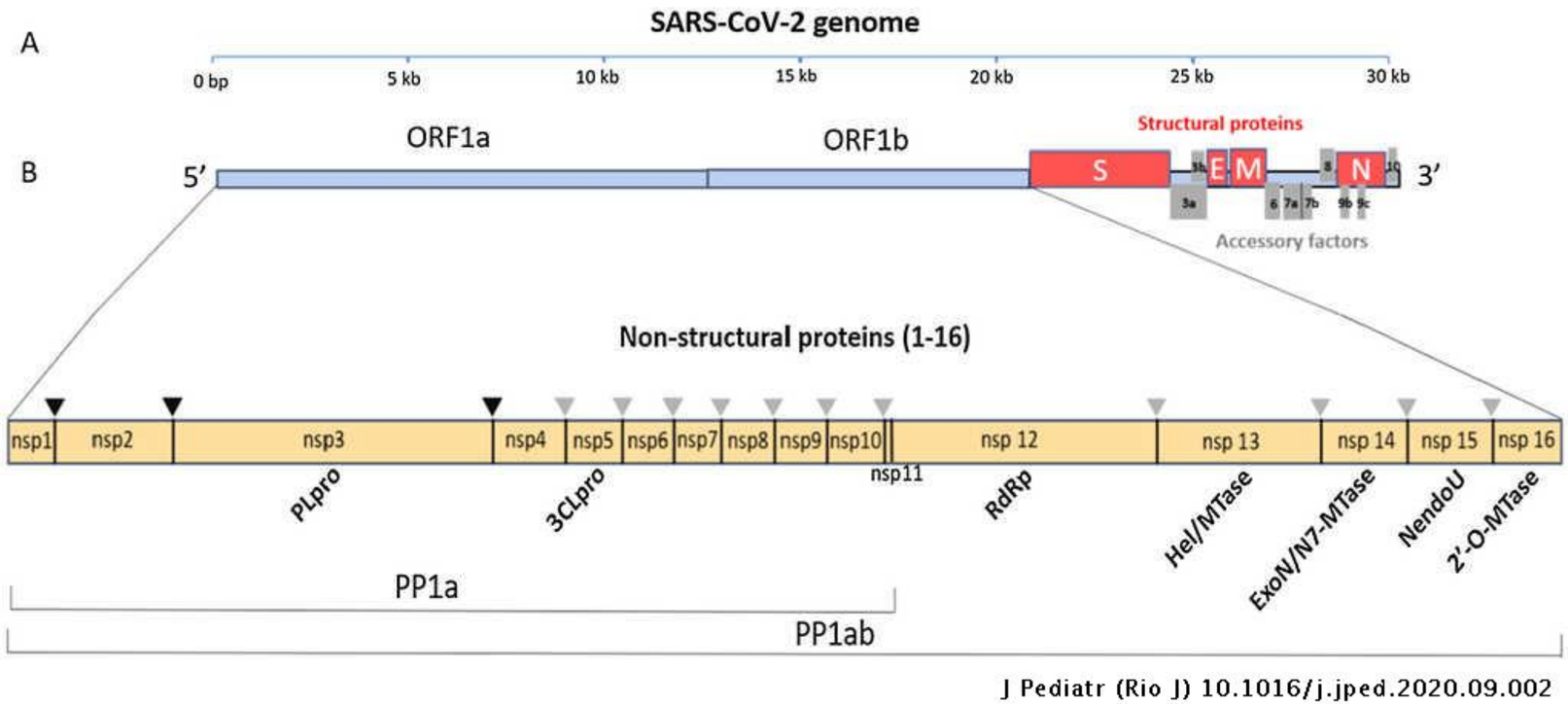




Sars-cov-2 has the largest RNA viral genome, which ranges from 26 to 32 kb. The genome of the virus starts with 5' carbon base, has an ORF1ab, S, E, M, N, and accesory region, and then end with 3' carbon base. ORF1ab are non-structural protiens. ORF1ab, open reading frames, comprises of 16 NSP regions. Nsp1 and nsp2 encode enzymes that bind to RNA and suppress gene expression. Nsp3 encodes enzymes that assist in the translation of viral mRNAs. Nsp4 encodes viroporin, which modify cellular membranes. Nsp5 encodes the organization of the viroplasm, where viral replication and assembly occur. Nsp6 generates autophagosomes, vesicles with cellular material that will be removed through autophagy. Nsp7 through 16 encode the generation of RNA synthesis and processing. Nsp7 through 11 encode the contribution to the nsp interactome, molecular interactions. Nsp12 encodes RNA polymerase. Nsp13 encodes helicase. Nsp14 through 16 encode mRNA capping, attaches the 5' cap to mRNA. The S region encodes the spike proteins, which bind to host cell receptors and fuse to the cell membrane. The M and E regions encode the membrane and envelope protiens, which assist in budding. The N region encodes the nucleocapsid protein, which assigns the viral replication-transcription complexes. The genome additionally contains an ORF coding for accessory proteins, which are not essential in virus replication but assist in pathogenesis, development of the virus.

In [13]:
```
Image(filename='SARS-COV-2_Genome.jpeg')
```

Out[13]:

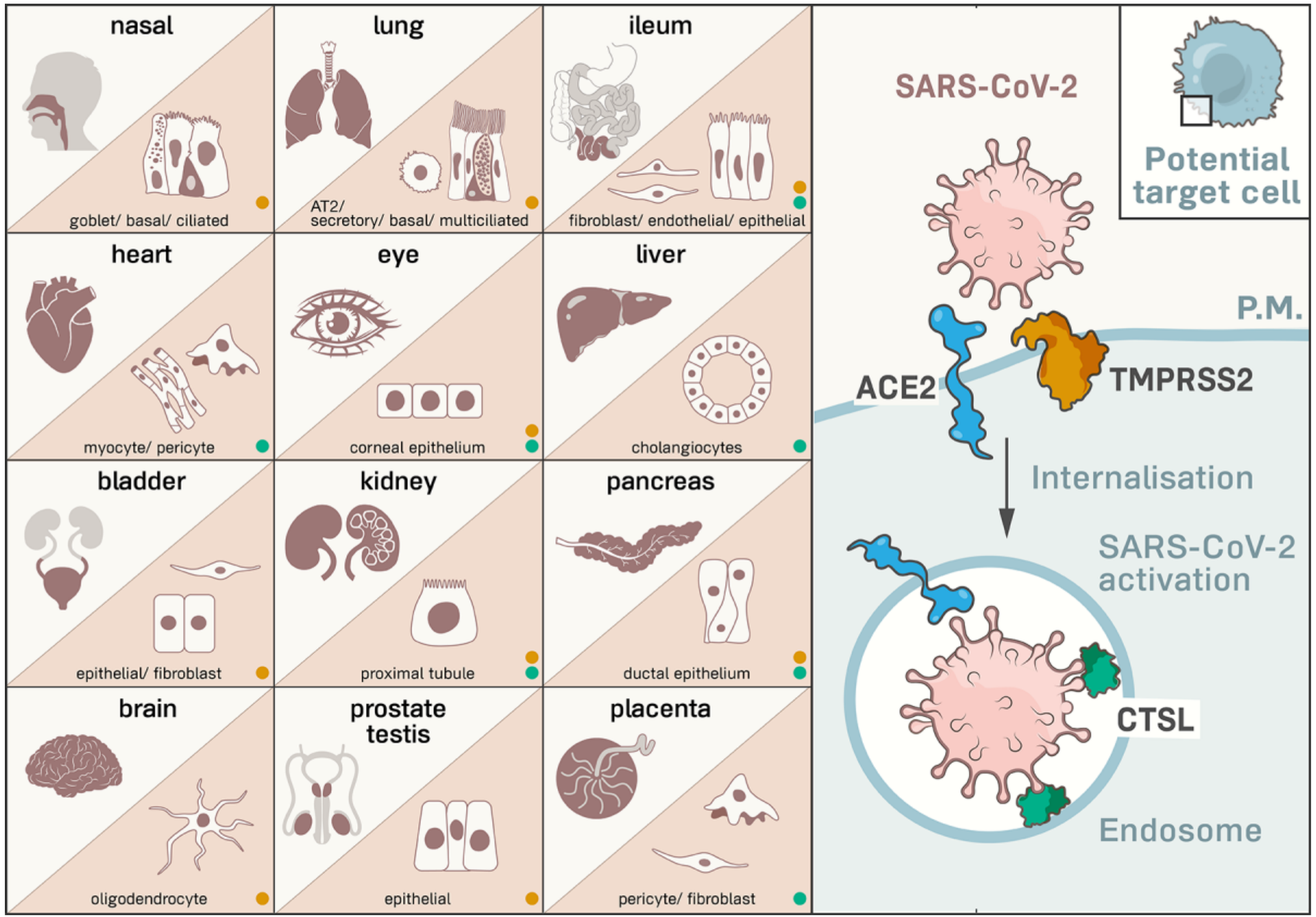


J Pediatr (Rio J) 10.1016/j.jped.2020.09.002

ACE2 modulates the activity of the angiotensin II protein, which increases blood pressure and inflammation. The SARS-CoV-2 virus binding to ACE2 inhibits ACE2 and can cause an increase in cell inflammation resulting in the death of cells in the alveoli units with hypoxic fluid flooding, an increase in blood preasure damaging blood vessels causing microvascular thrombosis, and the tranduction of the kidney podocytes resulting in acute kidney injury. The following is a diagram of the enzymes that get inhibited and the parts of the body that can get damaged.

In [5]:
```
Image(filename='ace2-thumb-l.png')
```

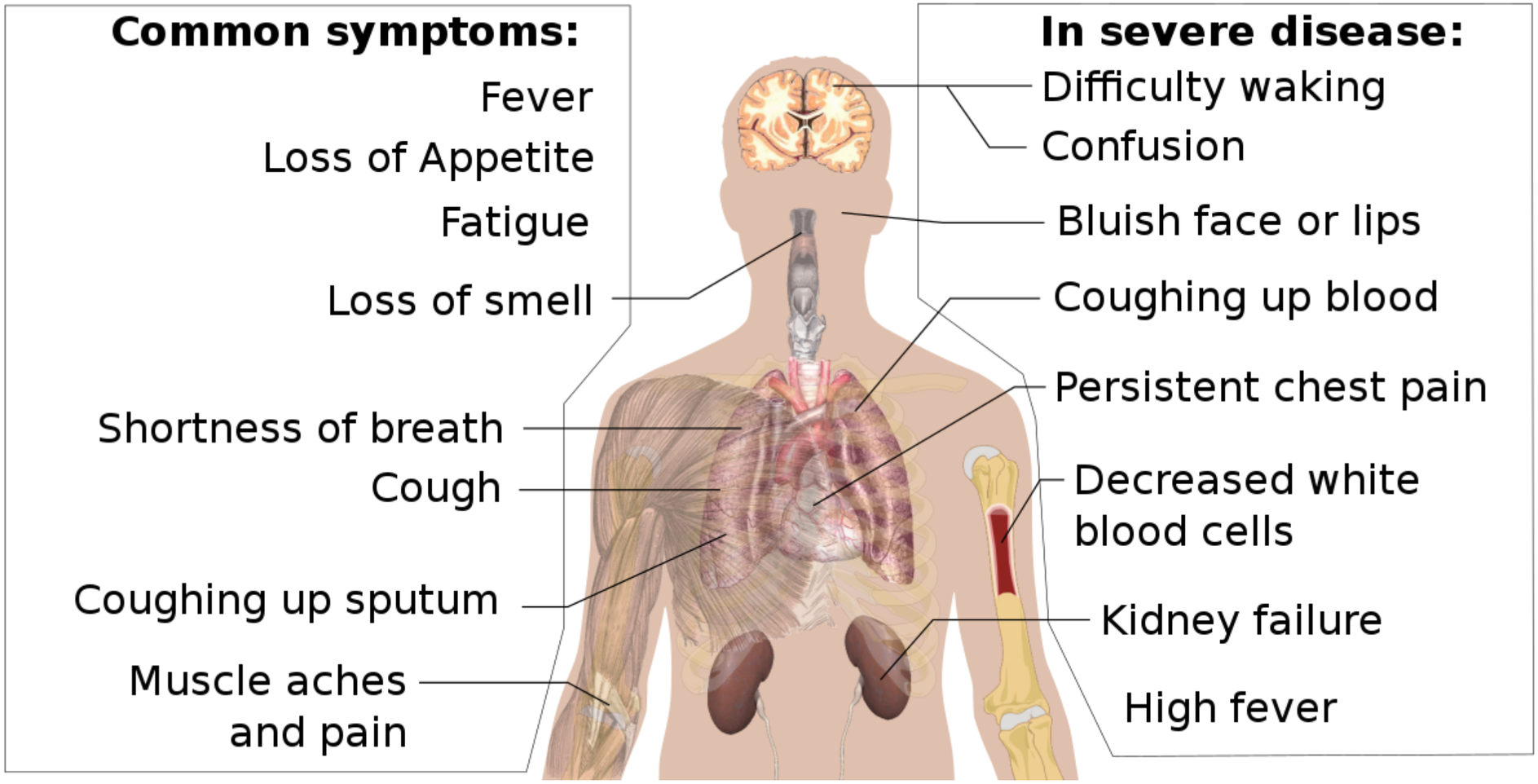
Out[5]:



SARS-CoV-2 is spread when an infected person coughs or sneezes droplets of saliva or mucus with the virus into the air. The respiratory droplets usually do not go further than a few feet, are airbourne for a few moments, and then land on a surface. SARS-CoV-2 has an incubation period of 2 to 14 days and the symptoms of the virus include cough, fever or chills, shortness of breath or difficulty breathing, muscle or body aches, sore throat, loss of taste or smell, diarrhea, headache, fatigue, nausea or vomiting, congestion or runny nose, and in rare cases the virus can lead to difficulty walking, confusion, bluish face or lips, coughing up blood, severe respiratory problems, kidney failure, high fever, or death. The following is a diagram of the symptoms.

In [4]:
```
Image(filename='Symptoms_of_coronavirus_disease_2019_4.0.svg.png')
```

Out[4]:

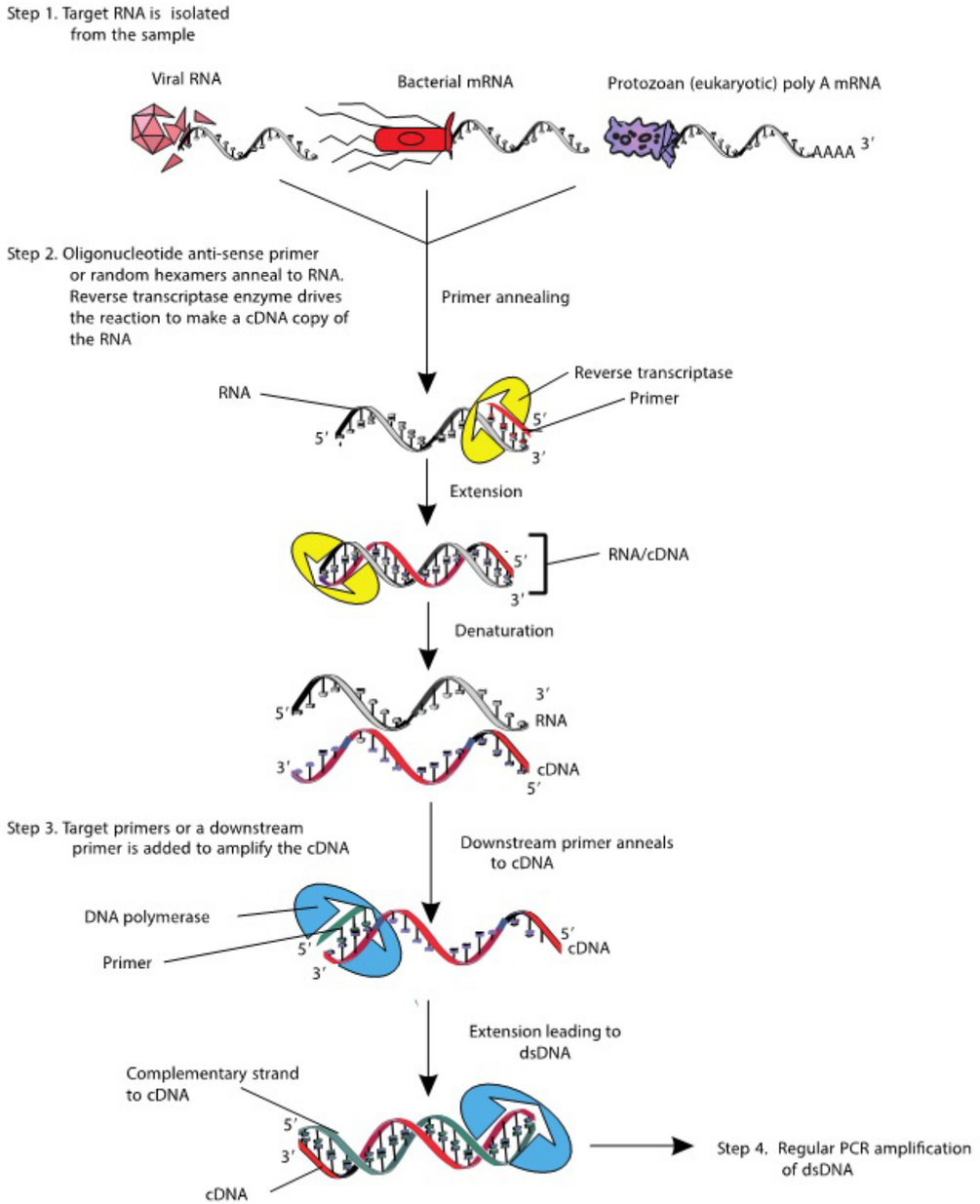


Testing for SARS-CoV-2

A patient can be tested for SARS-CoV-2 with the sequence-specific molecular nucleic acid assay, or the antigen-specific immunoassay. For a molecular-assay, a nasal or saliva sample is collected from upper respiratory fluid and then Real-time RT-PCR is conducted to quantify sequences within the RNA samples. Reverse transcriptase converts extracted RNA into cDNA that is used as a template for DNA polymerase to complete the strand of dsDNA and then RT-PCR amplifies the genetic regions and fluorescent probes bind to the regions for identification. For an immuno-assay, antibodies are put on a membrane and complexed with a potentialy virulent sample of which any antigen will be trapped that results in the membrane changing color. A SARS-CoV-2 recovered patient can be tested for whether the patient has developed anibodies against the virus by checking a blood sample for the antibodies. The following are diagrams of a molecular assay

In [6]:
```
Image(filename='10104fig1.jpg')
```

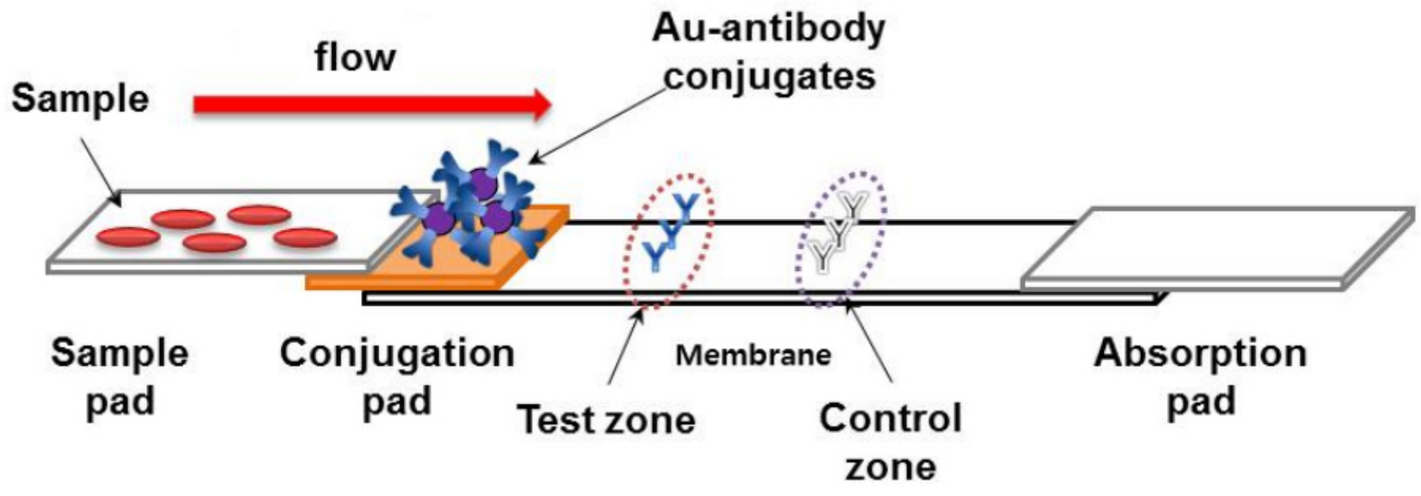
Out[6]:



and an immuno assay.

In [7]:
```
Image(filename='tag-lateral-flow-immunoassay-2.jpg')
```

Out[7]:



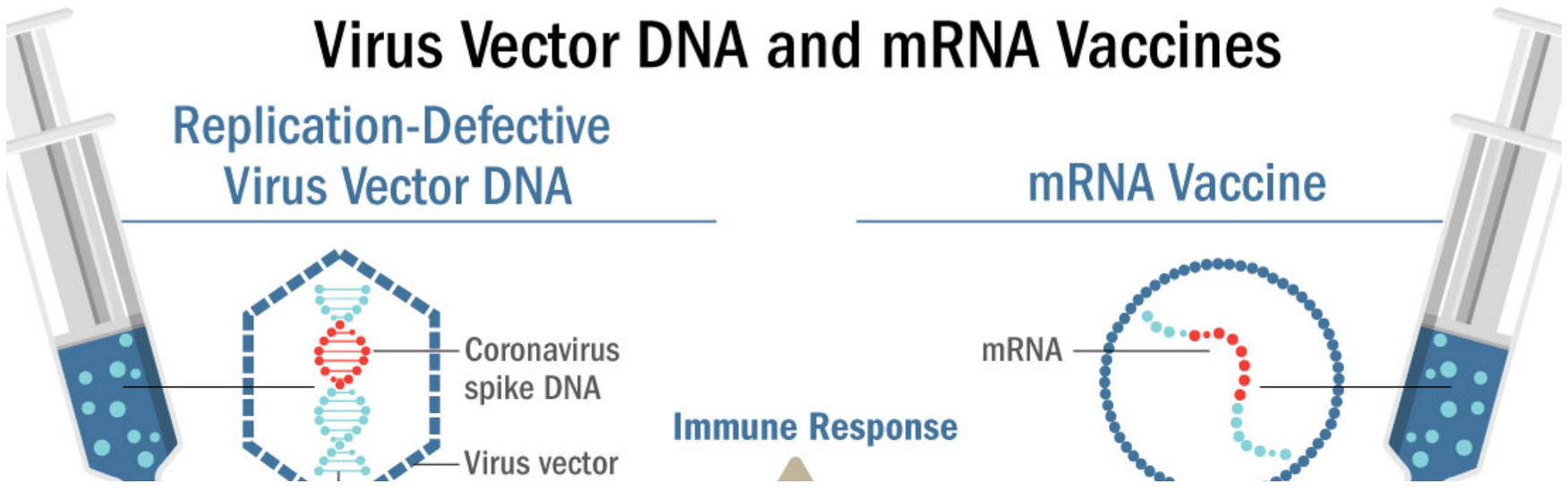
Vaccines for SARS-CoV-2

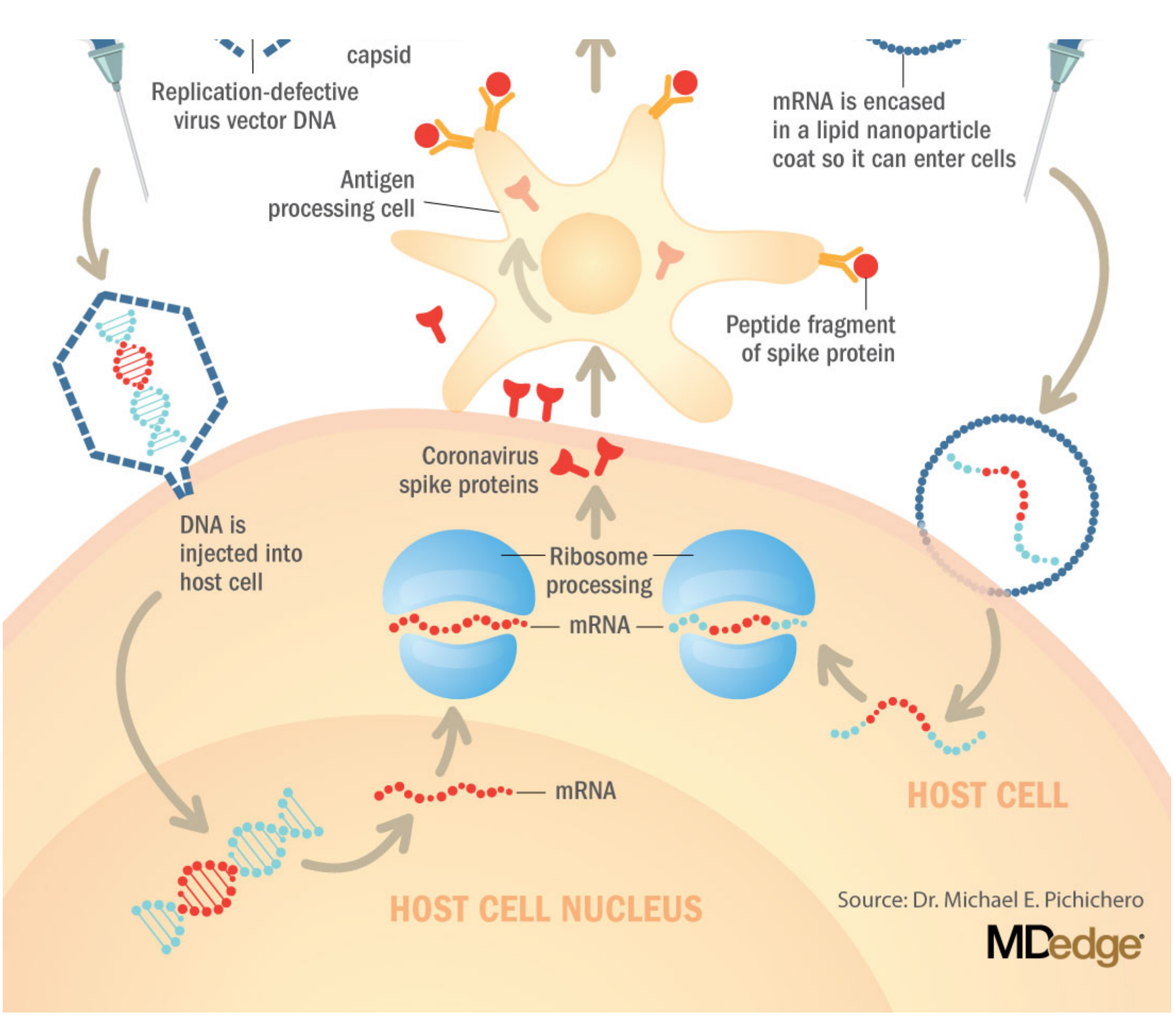
Vaccines have been developed to combat SARS-CoV-2. The Pfizer/BioNTech vaccine (BNT162b2) and The ModernaTX, Inc vaccine (mRNA-1273) are mRNA vaccines. mRNA vaccines deliver lab designed viral mRNA made by coding both the 5′-untranslated region (UTR) and the 3′-UTR into cells where it is translated into the encoded antigen to which an immune recognition by white blood cells, immunogenicity, results. The AstraZeneca in collaboration with the University of Oxford vaccine (AZD1222) is an adenovirus vaccine. Adenovirus vaccines are first gene sub-cloned into an intermediary vector, transfered to an adenovirus recombinant genome vector, transfected into packaging cells, amplified into a culture stock, and then titrated to determine the concentration of active adenoviruses in the stock. The adenoviruse is then delivered to cells where immunogenicity results. The following is a diagram of Virus vector DNA and mRNA vaccine immune response production.

The following is a diagram of Virus vector DNA and mRNA vaccine immune response production. Vaccines have been developed to combat SARS-CoV-2. The Pfizer/BioNTech vaccine (BNT162b2) and The ModernaTX, Inc vaccine (mRNA-1273) are mRNA vaccines. mRNA vaccines deliver lab designed viral mRNA made by coding both the 5′-untranslated region (UTR) and the 3′-UTR into cells where it is translated into the encoded antigen to which an immune recognition by white blood cells, immunogenicity, results. The AstraZeneca in collaboration with the University of Oxford vaccine (AZD1222) is an adenovirus vaccine. Adenovirus vaccines are first gene sub-cloned into an intermediary vector, transfered to an adenovirus recombinant genome vector, transfected into packaging cells, amplified into a culture stock, and then titrated to determine the concentration of active adenoviruses in the stock. The adenoviruse is then delivered to cells where immunogenicity results. The following is a diagram of Virus vector DNA and mRNA vaccine immune response production.

In [8]:
```
Image(filename='DNA_and_mrNA_Vaccines_web.jpeg')
```

Out[8]:





Import data of sars-cov-2 genome.

```
df=Json('sars-cov-2_genome')
```

In [52]:

```python
df.file("genome.fna")
genome=json_storage['sars-cov-2_genome']
genome
```

Out[52]:

```
'\nACUUUGUGUCUACUCCUCUCAACUAAACGAAAUUUUUCUAGUGCUGUCAUUUGUUAUGGCAGUCCUAGUGUAAUUGAAAUUUCGUCAAGUUUGUAAACUGGUUAGGCAAGUGUUGUAUUUUCUGUGUUUAAGCACUGGUGGUUCUGUCCACUAGUGCACACAUUGAUACUUAAGUGGUGUUC
UGUCACUGCUUAUUGUGGAAGCAACGUUCUGUCGUUGUGGAAACCAAUAACUGCUAACCAUGUUUUACAAUCAAGUGACACUUGCUGUUGCAAGUGAUUCGGAAAUUUCAGGUUUUGGUUUUGCCAUUCCUUCUGUAGCCGUUCGCGCUUAUAGCGAAGCCGCUGCACAAGGUUUUCAGGCAUGC
CGCUUUGUUGCUUUUGGCUUACAGGAUUGUGUAACCGGUAUUAAUGAUGACGAUUAUGUCAUUGCAUUGACUGGUACUAAUCAGCUUUGUGCCAAAAUUUUACUUUUUUCUGAUAGACCUCUUAAUUUGCGAGGUUGGCUCAUUUUUUCUAACAGCAAUUAUGUUCUUCAGGACUUUGAUGUUGU
UUUUGGCCAUGGUGCAGGAAGUGUGGUUUUUGUGGAUAAGUAUAUGUGUGGUUUUGAUGGUAAACCUGUGUUACCUAAAAACAUGUGGGAAUUUAGAGAUUACUUUAAUGAUAAUACUGAUAGUAUUGUUAUUGAUGGUGUCACUUAUCAAUUAGCAUGGGAUGUUAUACGUAAAGACCUUUCUU
AUGAACAGCAAAAUGUUUUAGCUAUUGAGAGCAUUCAUUAUCUUGGUACUACAGGUCAUACUUUGAAGUCUGGUUGCAAACUCAUUAAUGCCAAGCCGCCUAAAUAUUCUUCUAAGGUUGUUUUGAGUGGUGAAUGGAAUGCUGUGUAUAAGGCGUUUGGUUCACCAUUUAUUACAAAUGGUAUA
UCAUUGCUAGAUAUAAUUGUUAAACCAGUUUUCUUUAAUGCUUUUGUUAAAUGCAAUUGUGGUUCUGAGAAUUGGAGUGUUGGUGCAUGGGAUGGUUAUCUAUCUUCUUGUUGUGGCACACCUGCUAAGAAACUUUGUGUUGUUCCUGGUAAUGUUGUUCCUGGUGAUGUGAUCAUCACCUCAAC
UGAUGCUGGUUGUGGUGUUAAAUACUAUGCUGGCUUAGUUGUUAAACAUAUUACUAACAUUACUGGUGUGUCUUUAUGGCGUGUUACAGCUGUUCAUUCUGAUGGAAUGUUUGUGGCAACAUCUUCUUAUGAUGCACUCUUGCAUAGAAAUUCAUUAGACCCUUUUUGCUUUGAUGUUAACACUU
UACUUUCUAAUCAAUUACGUCUAGCUUUUCUUGGUGCUUCUGUUACAGAAGAUGUUAAAUUUGCUGCUAGCACUGGUGUUAUUGACAUUAGUGCUGGUAUGUUUGGUCUUUACGAUGACAUAUUGACAAACAAUAAACCUUGGUUUGUACGCAAAGCUUCUGGGCUUUUUGAUGCAAUCUGGGAU
GCUUUUGUUGCCGCUAUUAAGCUUGUGCCAACUACUACUGGUGGUUUGGUUAGGUUUGUUAAGUCUAUCGCUUCAACUGUUUUAACUGUUUCUAAUGGGUGUUAUUAUUAUGUGUGCAGAUGUUCCAGAUGCUUUUCAACCAGUUUACCGUACAUUUACACAAGCCAUUUGUGCUGCAUUUGAUUU
UUCUUUAGAUGUAUUUAAAAUUGGUGAUGUUAAAUUUAAACGACUUGGUGAUUAUGUCUUACUGAAAAUGCUCUUGUUCGUUUGACUACUGAAGUUGUUCGUGGUGUUCGUGAUGCUCGCAUAAAGAAAGCCAUGUUUACUAAAGUAGUUGUAGGUCCUACAACUGAAGUUAAGUUUUCUGUUA
UUGAACUUGCCACUGUUAAUUUGCGUCUUGUUGAUUGUGCACCUGUAGUUUGCCCUAAAGGUAAAAUUGUUGUUAUUGCUGGACAAGCUUUUUUCUAUAGUGAUGGUUUUUAUCGUUUUAUGGUUGAUUCUACAACUGUAUUAAAUGACCCUGUUUUUACUGGUGAUUUAUUUUACACUAUUAAG
UUUAGUGGUUUUAAGCUUGAUGGUUUUAACCAUCAGUUUGUUAAUGCUAGUUCUGCUACAGAUGCCAUUAUUGCUGUUGAGCUGUUGUUAUCGGAUUUUAAAACUGCAGUUUUUGUGUACACAUGUGUGGUUGAUGGUUGUAGUGUCAUUGUUAGACGUGAUGCUACAUUCGCCACACAUGUGUG
UUUUAAGGACUGUUAUAAUAUUUGGGAGCAAUUCUGCAUUGAUAAUUGUGGUGAGCCAUGGUUUUUGACUGAUUAUAAUGCUAUCUUGCAGAGUAAUAACCCUCAAUGUGCUAUUGUUCAAGCAUCGGAGUCUAAAGUUUUGCUUGAGAGGUUUUUACCUAAGUGUCCUGAAAUACUGUUGAGUA
UUGAUGAUGGCCAUUUAUGGAAUCUUUUUGUUGAAAAGUUUAAUUUUGUUACAGAUUGGUUAAAAACUCUUAAGCUUACACUUACUUCUAAUGGUCUUUUAGGUAAUUGUGCCAAACGUUUUAGACGUGUUUUGGUAAAAUUGCUUGAUGUCUAUAAUGGUUUUCUUGAAACUGUCUGUAGUGUC
GUACACACUGCUGGUGUUUGCAUUAAAUAUUAUGCUGUUAAUGUUCCAUAUGUAGUUAUUAGUGGUUUUGUAAGUCGUGUAAUUCGUAGAGAAAGGUGUGACGUGACUUUUCCUUGUGUUAGUUGUGUCACUUUUUUCUAUGAAUUUUAGACACGUGUUUUAGUGUUAGUAAACCUAAUGCCAU
UGAUGUUGAACAUUUAGAGCUUAAAGAAACCGUUUUUGUUGAACCUAAGGAUGGUGGUCAAUUUUUUGUUUCUGAUGAUUAUCUUUGGUAUGUUGUAGAUGACAUUUAUUAUCCAGCUUCAUGUAAUGGUGUAUUGCCAGUUGCUUUUACAAAAUUGGCAGGUGGUAAAAUAUCUUUUUCUGAUG
AUGUUAUAGUCCAUGAUGUUGAACCUACCCAUAAAGUCAAGCUCAUAUUUGAGUUUGAAGAUGAUGUUGUUACCAGUCUUUGUAAGAAGAGUUUUGGUAAGUCUAUUAUUUUAUACAGGUGAUUGGGAAGGUUUACAUGAAGUUCUUACAUCUGCAAUGAAUGUCAUUGGGCAACAUAUUAAGUUG
CCACAAUUUUAUAUUUAUGAUGAAGAGGGUGGUUAUGAUGUUUCUAAACCAGUUAUGAUUUCACAAUGGCCUAUUAGUGAUGAUAGUGGUGAUUGUGUUGUUGAAGCGAGCACUGAUUUUCAUCAAUUAGAAUUUGUUAGAGAAGAGGUUGAUAUAAUUGAACAACCUUUUGGGGAAGUUGAACA
UUCGCUCCCAAUUAAACAACCUUUUUCUUUUUCUUUUAGAGAUGAAUUGGGUGUUCGUGUUUUAGAUCAAUCUGAUAAUAAUUGUUGGAUUAGUACCACACUUAUACAGUUGCAACUUACAAAGCUUUUGGAUGAUUCUAUUGAGAUGCAAUUGUUUAAAGUUGGUAAAGUUGAUUCAAUUGUUC
AAAAGUGUUAUGAGUUGUCUCAUUUAAUUAGUGGUUCACUUGGUGAUAGUGGUAAACUUCUUAGUGAACUUCUUAAAGAUAAAUAUACAUGUUCUAUAACUUUUGAGAUGUCUUGUGAUUGUGGUAAAAAGUUUGAUGAGCAAGUUGGUUGUUUGUUUUGGAUUAUGCCUUACACAAAACUUUUU
CAAAAAGGUGAGUGUUGUAUUUGUCAUAAAAUGCAGACUUAUAAGCUUGUUAGUAUGAAAGGUACUGGUGUGUUUGUACAGGAUCCAGCACCUAUUGACAUUGAUGCUUUCCCUGUGAGACCUAUAUGUUCAUCUGUAUAUUUAGGUGUUAAGGGUUCUGGUCAUUAUCAAACAAAUUUAUACAG
UUUUGACAAAGCUAUUGAUGGUUUUGGUGUCUUUGACAUUAAAAAUAGUAGUGUUAAUACUGUCUGUUUUGUUGAUGUUGAUUUUCAUAGUGUAGAAAUAGAAGCUGGUGAAGUUAAACCUUUUGCUGUAUAUAAAAAUGUUAAAUUUUAUUUAGGUGAUAUUUCACACCUUGUAAACUGUGUUU
CUUUUGACUUUGUUGUCAAUGCUGCUAAUGAAAAUCUCAUGCAUGGAGGCGGUGUUGCACGUGCUAUUGAUAUUUUGACUGAAGGUCAACUUCAGUCAUUAUCUAAAGAUUACAUUAGUAGUAAUGGUCCACUUAAGGUUGGAGCAGGUGUUAUGUUGGAGUGUGAAAAAUUCAAUGUAUUUAAU
GUUGUUGGUCCGCGAACUGGUAAACAUGAGCAUUCAUUACUUGUUGAAGCUUAUAAUUCUAUUUUAUUUGAAAAUGGUAUUCCACUUAUGCCUCUUCUUAGUUGUGGUAUUUUUGGUGUAAGGAUUGAAAAUUCUCUUAAAGCUUUGUUUAGUUGUGACAUUAAUAAACCGUUGCAAGUUUUUGU
UUAUUCUUCAAAUGAAGAACAAGCUGUUCUUAAGUUUUUAGAUGGUUUAGAUUUAACACCAGUCAUUGAUGAUGUUGAUGUUGUUAAACCUUUUAGAGUUGAAGGUAAUUUUUCAUUCUUUGAUUGUGGUGUCAAUGCCUUGGAUGGUGAUAUUUACUUAUUAUUUACUAACUCUAUUUUAAUGU
UGGAUAAACAAGGACAAUUAUUGGACACAAAACUUAAUGGUAUUUUGCAACAGGCAGCUCUUGAUUAUCUUGCUACAGUUAAAACUGUACCAGCUGGUAAUUUGGUUAAACUUGUUGUUGAGAGUUGUACCAUUUAUAUGUGUGUUGUACCAUCGAUAAAUGAUCUUUCUUUUGAUAAAAAUCUU
GGUCGUUGUGUGCGUAAACUUAAUAGAUUGAAAACUUGUGUUAUUGCCAAUGUUCCUGCUAUUGAUGUUUUGAAAAAGCUUCUUUCAAGUUUGACUUUAACUGUUAAAUUUGUUGUAGAGAGUAAUGUUAUGGAUGUUAACGACUGUUUUAAGAAUGAUAAUGUAGUUUGAAAAUCACUGAAGA
UGGUAUUAAUGUUAAAGAUGUUGUUGUUGAGUCUUCUAAGUCACUUGGUAAACAAUUGGGUGUUGUGAGUGAUGGUGUUGACUCUUUUGAAGGUGUUUUACCUAUUAAUACUGAUACUGUCUUAUCUGUAGCUCCAGAAGUUGACUGGGUUGCUUUUUACGGUUUUGAAAAGGCAGCACUUUUUG
CUUCUUUGGAUGUAAAGCCAUAUGGUUACCCUAAUGAUUUUGUUGGUGGUUUUAGAGUUCUUGGGACCACCGACAAUAAUUGUUGGGUUAAUGCAACUUGUAUAAUUUUACAGUAUCUUAAGCCUACUUUUAAAUCUAAGGGUUUAAAUGUUCUUUGGAAUAAAUUUGUUACAGGUGAUGUUGGA
CCUUUUGUUAGUUUUAUUUAUUUUAUAACUAUGUCUUCAAAGGGUCAAAAGGGUGAUGCUGAAGAGGCAUUAUCUAAAUUGUCAGAGUAUUUGAUUAGUGAUUCUAUUGUUACUCUUGAACAAUAUUCAACUUGUGACAUUUGUAAAAGUACUGUAGUUGAAGUUAAAAGUGCUAUUGUCUGUGC
UAGUGUGCUUAAAGAUGGUUGUGAUGUUGGUUUUUGUCCACACAGACAUAAAUUGCGUUCACGUGUUAAGUUUGUUAAUGGACGUGUUGUUAUUACCAAUGUUGGUGAACCUAUAAUUUCACAACCUUCUAAGUUGCUUAAUGGUAUUGCUUAUACAACAUUUUCAGGUUCUUUUGAUAACGGUC
ACUAUGUAGUUUAUGAUGCUGCUAAUAAUGCUGUCUAUGAUGGUGCUCGUUUAUUUGCUUCAGAUUUGUCUACUUUAGCUGUUACAGCUAUUGUUGUAGUAGGUGGUUGUGUAACAUCUAAUGUUCCAUCAAUUGUUAGUGAGAAAAUUUCUGUUAUGGAUAAACUUGAUACUGGUGCACAAAAA
UUUUUCCAAUUUGGUGAUUUUGUUAUGAAUAACAUUGUUCUGUUUUUAACUUGGUUGCUUAGUAUGUUUAGUCUUUUACGUACUUCUAUUAUGAAGCAUGAUAUUAAAGUUAUUGCCAAGGCUCCUAAACGUACAGGUGUUAUUUUGACACGUAGUUUUAAGUAUAACAUUAGAUCUGCUUUGUU
UGUUAUAAAGCAGAAGUGGUGUGUUAUUGUUACUUUGUUUAAGUUCUUAUUGUUAUUAUAUGCUAUUUAUGCACUUGUUUUUAUGAUUGUGCAAUUUAGUCCUUUUAAUAGUCUUUUAUGUGGUGACAUUGUAAGUGGUUAUGAAAAAUCCACUUUUAAUAAGGAUAUUUAUUGUGGUAAUUCUA
UGGUUUGUAAGAUGUGUUUGUUCAGUUAUCAAGAGUUUAAUGAUUUGGAUCAUACUAGUCUUGUUUGGAAGCACAUUCGUGAUCCUAUAUUAAUCAGUUUACAACCAUUUGUAUACUUGUUAUUUUGUUAAUUUUUGGUAAUAUGUAUUUGCGUUUUGGACUUUUAUAUUUUGUUGCACAAUUU
AUUAGUACUUUUGGUUCUUUCUUAGGCUUUCAUCAGAAACAGUGGUUUUUACAUUUUGUGCCGUUUGAUGUUUUAUGUAAUGAGUUUUUAGCUACAUUUAUUGUCUGCAAAAUUGUUUUAUUUGUUAGACAUAUUAUUGUUGGCUGUAAUAAUGCUGACUGUGUAGCUUGUUCUAAAAGUGCUAG
ACUUAAACGUGUACCACUUCAAACUAUUAUUAAUGGUAUGCAUAAAUCAUUCUAUGUUAAUGCUAAUGGUGGUACUUGUUUCUGUAAUAAACAUAACUUCUUUUGUGUUAAUUGUGAUUCUUUUGGGCCUGGUAACACUUUUAUUAAUGGUGAUAUUGCAAGAGAGCUUGGUAAUGUUGUUAAAA
CAGCUGUUCAACCCACAGCUCCUGCAUAUAUUAUUAUUGAUAAGGUAGAUUUUGUUAAUGGAUUUUAUCGUCUUUAUAGUGGUGACACUUUUUGGCGGUAUGACUUUGACAUUACUGAAUCUAAGUAUAGUUGUAAAGAGGUUCUGAAGAAUUGUAAUGUUUUAGAAAAUUUUAUUGUUUACAAU
AAUAGUGGUAGUAACAUUACACAGAUUAAAAAUGCUUGUGUUUAUUUUUCUCAAUUGUUGUGUGAACCUAUAAAGUUGGUAAAUUCAGAGUUGUUGUCAACUUUAUCAGUUGAUUUUAAUGGUGUUUUGCAUAAGGCAUAUGUUGAUGUUUUGUGUAAUAGUUUUUUUAAGGAGCUAACUGCUAA
CAUGUCCAUGGCUGAAUGUAAAGCUACACUUGGUUUGACUGUUUCUGAUGAUGAUUUUGUUUCAGCUGUUGCCAAUGCACAUAGGUAUGACGUUUUGCUUUCAGAUUUGUCAUUUAAUAAUUUUUUUAUUUCUUAUGCUAAACCUGAAGAUAAGUUGUCCGUUUAUGAUAUUGCUUGUUGAUGC
GUGCCGGUUCUAAGGUUGUUAACCAUAAUGUUUUAAUUAAAGAGUCAAUACCUAUUGUUUGGGGUGUCAAGGACUUUAAUACUCUUUCUCAAGAAGGUAAGAAGUACCUUGUUAAAACAACUAAAGCAAAGGGUUUGACUUUUUUAUUAACUUUUAAUGAUAACCAAGCAAUUACACAAGUUCCU
GCUACUAGUAUAGUUGCAAAACAGGGUGCUGGUUUUAAACGUACUUAUAAUUUUCUGUGGUAUGUAUGUUUAUUUGUUGUUGCAUUGUUUAUUGGUGUCUCAUUUAUUGAUUACACAACCACUGUAACUAGCUUUCAUGGUUAUGAUUUUAAGUACAUUGAGAAUGGUCAGUUGAAGGUGUUUGA
AGCACCUUUACACUGUGUUCGUAAUGUUUUUGAUAAUUUUAAUCAAUGGCAUGAGGCUAAGUUUGGUGUGUUGUUACUACUAAUAGUGAUAAAUGUCCUAUAGUUGUUGGUGUUUCAGAGCGUAUUAAUGUUGUUCCUGGUGUUCCAACAAAUGUAUAUUUGGUAGGAAAGACUCUUGUUUUUACAU
UACAGGCUGCUUUUGGAAACACAGGUGUUUGUUAUGACUUUGAUGGUGUUACCACUAGUGAUAAGUGUAUUUUUAAUUCUGCUUGUACUAGGUUGGAAGGUUUGGGUGGUGACAAUGUUUAUUGUUACAACACUGAUCUUAUUGAAGGUUCUAAACCUUAUAGUACUUUACAGCCCAAUGCUUAU
UAUAAGUAUGAUGUUAAAAAUUAUGUACGUUUUCCAGAAAUUUUAGCUAGAGGUUUUGGCUUACGUACUAUUAGAACUUUGGCUACACGUUAUUGUAGAGUUGGUGAAUGCCGUGACUCACAUAAAGGUGUUUGUUUUGGUUUUGAUAAAUGGUAUGUUAAUGAUGGACGUGUUGAUGAUGGUUA
CAUUUGUGGUGAUGGUCUUAUAGACCUUCUUGUUAAUGUACUCUCAAUCUUUAGUUCAUCUUUUAGCGUUGUGGCUAUGUCUGGACAUAUGUUGUUUAACUUUCUUUUUGCAGCAUUUAUUACAUUUUUGUGCUUUUUAGUUACUAAAUUUAAACGUGUUUUUGGUGAUCUUUCUUAUGGUGUUU
UUACUGUUGUUUGUGCAACUUUGAUUAAUAACAUUUCUUAUGUUGUUACUCAAAAUUUAUUUUUUAUGUUGCUUUAUGCUAUUUUGUAUUUUGUUUUUACUAGGACAGUGCGUUAUGCUUGGAUUUGGCAUAUUGCAUACAUUGUUGCAUACUUCUUGUUAAUACCAUGGUGGCUUCUCACAUGG
UUUAGUUUUGCUGCAUUUUUAGAGCUUUUACCUAAUGUUUUUAAGUUAAAAAUCUCUACUCAAUUGUUUGAAGGUGAUAAGUUUAUAGGUACUUUUGAGAGUGCUGCUGCAGGUACAUUUGUUCUUGACAUGCGUUCUUAUGAAAGGCUGAUAAAUACUAUUUCACCUGAGAAACUUAAGAAUUA
UGCUGCAAGUUAUAAUAAAUAUAAAUAUUAUAGUGGUAGUGCUAGUGAGGCUGAUUAUCGUUGUGCUUGUUAUGCUCAUUUAGCCAAGGCUAUGUUAGAUUACGCAAAAGAUCAUAAUGACAUGUUAUAUUCUCCACCUACCAUUAGCUACAAUUCCACCUUACAAUCUGGUCUUAAGAAGAUGG
CACAACCAUCUGGUUGUGUUGAGAGAUGUGUGGUUCGCGUCUGUUAUGGUAGUACUGUGCUUAAUGGAGUUUGGUUAGGUGACACUGUUACUUGUCCUAGACAUGUCAUAGCACCAUCAACCACUGUUCUUAUUGAUUAUGAUCAUGCAUAUAGUACUAUGCGUUUGCAUAAUUUUUCAGUGUCU
CAUAAUGGUGUCUUCUUGGGAGUUGUUGGUGUUACAAUGCAUGGUUCUGUGUUGCGUAUUAAGGUUUCACAAUCUAAUGUACAUACACCUAAACAUGUUUUUAAAACGUUGAAACCUGGUGAUUCUUUUAAUAUUUUAGCAUGUUAUGAAGGUAUUGCAUCUGGUGUUUUGGUGUUAAUUUACG
UACAAACUUUACUAUUAAAGGUUCUUUUAUAAAUGGAGCUUGUGGUUCUCCUGGUUAUAAUGUUAGAAAUGAUGGUACUGUUGAGUUUUGUUAUUUACACCAAAUUGAGUUAGGUAGUGGUGCUCAUGUUGGUUCUGAUUUUACUGGUAGUGUUUAUGGUAAUUUUGAUGACCAACCUAGUUUGC
AAGUUGAGAGUGCCAACCUUAUGCUAUCAGAUAAUGUUGUUGCCUUUUUGUAUGCUGCUUUGUUGAAUGGUUGUAGGUGGUGGUUGCGUUCAACUAGAGUUAAUGUUGAUGGUUUUAAUGAAUGGGCUAUGGCUAAUGGUUAUACAAGUGUUUCUAGUGUUGAGUGCUAUUCUAUUUUGGCAGCA
AAAACUGGUGUUAGUGUUGAACAAUUGUUAGCUUCCAUUCAACAUCUUCAUGAAGGUUUUGGUGGUAAAAACAUACUUGGUUAUUCUAGUUUAUGUGAUGAGUUCACACUAGCUGAAGUUGUGAAGCAGAUGUAUGGUGUUAACUUGCAAAGUGGUAAGGUUAUUUUUGGUUUAAAAACAAUGUU
UUUAUUUAGCGUUUUCUUCACAAUGUUUUGGGCAGAACUCUUUAUUUAUACAAACACUAUAUGGAUAAACCCUGUGAUACUUACACCUAUAUUUUGUCUACUUUUGUUUUUGUCAUUAGUUUAACUAUGUUUCUUAAACAUAAGUUUUUGUUUUUGCAAGUAUUUUUAUUACCUACUGUUAUUG
CAACUGCUUUAUAUAAUUGUGUUUUGGAUUAUUACAUAGUAAAAUUUUUGGCUGACCAUUUUAACUAUAAUGUUUCAGUAUUACAAAUGGAUGUUCAGGGUUUAGUUAAUGUUUUGGUUGUUUAUUUGUUGUAUUUUUACACACAUGGCGCUUUUCUAAAGAACGUUUUACACAUUGGUUUACA
UAUGUGUGUUCUCUUAUAGCAGUAGCUUACACUUAUUUUUAUAGUGGUGACUUUUUGAGUUUGCUUGUUAUGUUUUUAUGUGCUAUAUCUAGUGAUUGGUACAUUGGUGCCAUUGUUUUUAGGUUGUCACGUUUGAUUGUAAUUUUUUCACCUGAAAGUGUAUUUAGUGUUUUUGGUGAUGUGAA
ACUUACUUUAGUUGUUUAUUUAAUUUGUGGUUAUUUAGUUUGUACUUAUUGGGGCAUUUUGUAUUGGUUCAAUAGGUUUUUUAAAUGUACUAUGGGUGUUUAUGAUUUUAAGGUGAGUGCUGCUGAAUUUAAAUACAUGGUUGCUAAUGGACUUCAUGCACCACAUGGACCUUUUGAUGCACUUU
GGUUAUCAUUCAAACUACUUGGUAUUGGUGGUGACCGUUGUAUAAAAAUUUCAACUGUCCAAUCCAAACUGACUGAUUUGAAGUGUACUAAUGUUGUGUUAUUGGGUUGUUUGUCUAGUAUGAACAUUGCAGCUAAUUCUAGUGAAUGGGCUUAUUGUGUUGAUUUACACAAUAAGAUUAAUCUU
UGUGAUGACCCUGAAAAAGCUCAAGGUAUGUGUUAGCACUCCUUGCGUUCUUUCUAAGUAAACAUAGUGAUUUUGGUCUUGAUGGCCUUAUUGAUUCUUAUUUUGAUAAUAGUAGCACCCUUCAGAGUGUUGCUUCAUCAUUUGUUAGUAUGCCAUCAUAUAUUGCUUAUGAAAAUGCUAGACA
AGCUUAUGAAGAUGCUAUUGCUAAUGGAUCUUCUUCUCAACUUAUUAAACAAUUGAAGCGUGCCAUGAAUAUCGCAAAGUCUGAAUUUGAUCAUGAGAUAUCUGUUCAGAAGAAAAUUAAUAGAAUGGCUGAACAAGCUGCUACACAGAUGUAUAAAGAAGCACGCUCUGUUAAUAGAAAAUCUA
```

AAGUUAUUAGUGCUAUGCACUCUUUACUUUUUGGAAUGUUAAGACGUUUGGAUAUGUCUAGUGUUGAAACUGUUUUGAAUUUAGCACGUGAUGGUGUUGUGCCAUUGUCAGUUAUACCUGCAACUUCAGCUUCCAAACUAACUAUUGUUAGUCCAGAUCUUGAAUCUUAUUCUAAGAUUGUUUGU
GAUGGUUCUGUUCAUUAUGCUGGAGUUGUUUGGACACUUAAUGAUGUUAAAGACAAUGAUGGUAGACCUGUUCAUGUUAAAGAGAUUACAAAGGAAAAUGUUGAAACUUUGACAUGGCCUCUUUAUCCUUAAUUGUGAACGUGUUGUUAAACUUCAAAAUAAUGAAAUUAUGCCUGGUAAACUUAA
GCAAAAACCUAUGAAAGCUGAGGGUGAUGGUGGUGUUUUAGGUGAUGGUAAUGCCUUGUAUAAUACUGAGGGUGGUAAAACUUUUAUGUACGCUUAUAUUUCUAAUAAAGCUGACCUUAAAUUUGUUAAGUGGGAGUAUGAGGGUGGUUGCAACACAAUCGAGUUAGACUCUCCUUGUCGAUUUA
UGGUCGAAACACCUAAUGGUCCUCAAGUGAAGUAUUUGUAUUUUGUUAAAAAUUUAAACACCUUACGUAGAGGUGCCGUUCUUGGUUUUAUAGGUGCCACAAUUCGUCUACAAGCUGGUAAACAAACUGAAUUGGCUGUUAAUUCUGGACUUUUAACUGCUUGUGCUUUUCUGUUGAUCCAGCA
ACUACUUACUUGGAAGCUGUUAAACAUGGUGCAAAACCUGUAAGUAAUUGUAUUAAGAUGUUAUCUAAUGGUGCUGGUAAUGGUCAAGCUAUAACAACUAGUGUAGAUGCUAACACCAAUCAAGAUUCUUAUGGUGGAGCGUCUAUUUGUUUGUAUUGUCGAGCUCACGUUCCUCACCCUAGUAU
GGAUGGUUACUGUAAGUUUAAGGGUAAAUGUGUUCAGGUCCUAUUGGUUGUUUGGAUCCUAUUAGGUUUUGUUAGAAAAUAAUGUGUGUAAUGUUUGUGGUUGGCUAUGGGGACAUGGAUGUGCUUGUGACCGUACAACUAUUCAAAGUUGUGACAUUUCUUAUUUAAACGAGCAAGGGGUUC
UAGUGCAGCUCGACUAGAACCCUGUAAUGGCACGGACAUCGAUAAGUGUGUCGUGCUUUUGACAUUUAUAAUAAAAAUGUUUCAUUCUUGGGUAAGUGUUUGAAGAUGAACUGUGUUCGUUUUAAAAAUGCUGAUCUUAAGGAUGGUUAUUUUGUUAUAAAGAGGUGUACUAAGUCGGUUAUGG
AACACGAGCAAUCCAUGUAUAACCUACUUAACUUUUCUGGUGCUUUGGCUGAGCAUGACUUCUUUACUUGGAAAGAUGGCAGAGUCAUUUAUGGUAAUGUUAGUAGACAUAAUCUUACUAAAUAUACUAUGAUGGACUUGGUCAUGCUAUGCUAUGCGUAACUUUGAUGAACAAAAUUGUGAUGUUCUA
AAAGAAGUAUUAGUUUUAACUGGUUGUUGUGACAAUUCUUAUUUUGAUAGUAAGGGUUGGUAUGACCCAGUUGAAAAUGAAGAUAUACAUAGAGUUUAUGCAUCUCUUGGCAAAAUUGUAGCUAGAGCUAUGCUUAAAUGCGUUGCUCUAUGUGAUGCGAUGGUUGCUAAAGGUGUUGUUGGUGU
UUUAACAUUAGAUAACCAAGAUCUUAAUGGUAACUUUUAUGAUUUUGGUGAUUUUGUUGUUAGCUUACCUAAUAUGGGUGUUCCCUGUUGUACAUCAUAUUAUUCUUAUAUGAUGCCUAUUAUGGGUUUAACUAAUUGUUUAGCUAGUGAGUGUUUUGUCAAGAGUGAUAUUUUUGGUAGUGAUU
UUAAAACUUUUGAUUUGCUUAAGUAUGAUUUCACUGAACAUAAAGAAAAUUAUUCAAUAAGUACUUUAAGCAUUGGAGUUUUGAUUAUCAUCCUAAUUGUAGUGACUGUUAUGAUGAUAUGUGUGUUAUACAUUGUGCUAAUUUUAAUACACUAUUUGCCACAACUAUACCAGGUACUGCUUUU
GGUCCACUAUGUCGUAAAGUUUUUAUAGAUGGUGUUCCACUUGUUACAACUGCUGGUUAUCAUUUUAAGCAAUUAGGUUUGGUUUGGAAUAAAGAUGUUAACACACACUCAGUUAGGUUGACAAUUACUGAACUUUUGCAAUUUGUCACCGACCCUUCCUUGAUAAUAGCUUCUUCCCCAGCACU
CGUUGAUCAACGCACUAUUUGUUUUUCUGUUGCAGCAUUGAGUACUGGUUUGACAAAUCAAGUUGUUAAGCCAGGUCAUUUUAAUGAAGAGUUUUAUAACUUUCUUCGUUUAAGAGGUUUCUUUGAUGAAGGUUCUGAACUUACAUUAAAACAUUUCUUCUUCGCACAGAAUGGUGAUGCUGCUG
UUAAAGAUUUUGACUUUUACCGUUAUAAUAAGCCUACCAUUUUAGAUAUUUGUCAAGCUAGAGUCACAUAUAAGAUAGUCUCUCGUUAUUUUGACAUUUAUGAAGGUGGCUGUAUUAAGGCAUGUGAAGUUGUUGUAACAAAUCUUAAUAAGAGUGCUGGUUGGCCAUUAAAUAAGUUUGGUAAA
GCUAGUUUGUAUUAUGAAUCUAUAUCUUAUGAAGAACAGGAUGCUUUGUUUGCUUUGACAAAGCGUAAUGUCUUCCCUACUAUGACACAGCUGAAUCUUAAGUAUGCUAUUAGUGGUAAAGAACGUGCUAGAACUGUUGGUGGUGUUUCUCUGUUGUCCACAAUGACCACAAGACAAUACCAUCA
AAAACAUCUUAAAUCCAUUGUUAAUACACGCAAUGCCACUGUUGUUAUUGGUACUACCAAAUUUUAUGGUGGUUGGAAUAAUAUGUUGCGUACUUUAAUUGAUGGUGUUGAAAACCCUAUGCUCAUGGGUUGGGAUUAUCCCAAAUGUGAUAGAGCUUUGCCUAACAUGAUACGUAUGAUUUCAG
CCAUGGUGUUGGGCUCUAAGCAUGUUAAUUGUUGUACUGCAACAGAUAGGUUUUAUAGGCUUGGUAAUGAGUUGGCACAAGUUUUAACAGAAGUUGUUUAUUCUAAUGGUGGUUUUUAUUUUAAGCCAGGUGGUACGACUUCUGGUGACGCUAGUACAGCUUAUGCUAAUUCUAUUUUUAACAUU
UUUCAAGCCGUAAGCUCUAACAUUAACAGGUUGCUUAGUGUCCCAUCAGAUUCAUGUAAUAAUGUUAAUGUUAGGGAUCUACAACGACGUCUGUAUGAUAAUUGCUAUAGGUUAACUAGUGUUGAAGAGUCAUUCAUUGAUGAUUAUUAUGGUUAUCUUAGGAAACAUUUUUCAAUGAUGAUUCU
CUCUGAUGAUGGUGUUGUCUGUUAUAACAAGGAUUAUGCUGAGUUAGGUUAUAUAGCAGACAUUAGUGCUUUUAAAGCCACUUUGUAUUACCAGAAUAAUGUCUUUAUGAGUACUUCUAAAUGUUGGGUUGAAGAAGAUUUAACUAAGGGACCACAUGAGUUUUGUUCCCAGCAUACUAUGCAAA
UAGUUGACAAAGAUGGUACCUAUUAUUUGCCUUACCCAGAUCCUAGUAGGAUCUUGUCAGCUGGUGUUUUUGUUGAUGAUGUUGUUAAGACAGAGCUGUGUUUUGUUAGAACGUUAUGUGCUUUAGCUAUUGAUGCAUACCCUCUUUCAAAACACCCUAAUUCUGAAUAUCGUAAGGUUUUU
UACGUAUUACUUGAUUGGGUUAAGCAUCUUAACAAAAAUUUGAAUGAGGGUGUUCUUGAAUCUUUUUCUGUUACACUUCUUGAUAAUCAAGAAGAUAAGUUUUGGUGUGAAGAUUUUUAUGCUAGUAUGUAUGAAAAUUCUACGAUAUUGCAAGCUGCUGGUUUAUGUGUUGUUUGGUUCACA
AACUGUACUUCGUUGUGGUGAUUGUCUGCGUAAGCCUAUGUUGUGCACUAAAUGCGCAUAUGAUCAUGUAUUUGGUACCGACCACAAGUUUAUUUUGGCUAUAACACCGUAUGUAUGUAAUGCAUCAGGUUGUGGUGUUAGUGAUGUCAAAAAAUUGUAUCUUGGUGGUUUGAAUUACUAUUGUA
CAAAUCAUAAACCACAGUUGUCUUUUCCAUUAUGUUCAGCUGGUAAUAUAUUUGGUUUAUAUAAAAAUUCAGCAACUGGUUCCUUAGAUGUUGAAGUUUUUAAUAGGCUUGCAACGUCUGAUUGGACUGAUGUUAGGACUAUAAACUUGCUAAUGAUGUUAAAGAUACACUUAGACUCUUUGCG
GCUGAAACUAUUAAAGCUAAAGAAGAGAGUGUUAAGUCUUCUUAUGCUUUUGCAACUCUUAAAGAGGUUGUUGGACCUAAAGAAUUGCUUCUUAGUUGGGAAAGUGGUAAAGUUAAACCACCUUUGAAUCGUAAUUCUGUUUUCACCUGUUUUCAAAUAAGUAAGGACUCAAAUUCCAAAUAGG
UGAGUUCAUCUUUGAGAAGGUUGAAUAUGGUUCUGAUACUGUUACGUAUAAGUCUACUGUAACUACUAAGUUAGUUCCUGGUAUGAUUUUUGUCUUAACAUCUCACAAUGUCCAACCUUUACGUGCACCAACUAUUGCUAACCAAGAGAAGUAUUCUAGCAUUUAUAAAUUGCACCCUGCUUUUA
AUGUCAGUGAUGCAUAUGCUAAUUUGGUUCCAUAUUACCAACUUAUUGGUAAACAAAAGAUAACUACAAUACAGGGUCCUCCUGGUAGUGGUAAGUCACAUUGUUCUAUUGGACUUGGAUUGUACUACCCAGGUGCGCGUAUUGUUUUUGUUGCUUGUGCCCAUGCUGCUGUUGAUUCCUUAUGU
GCAAAAGCUAUGACUGUUUAUAGCAUUGAUAAGUGUACUAGGAUUAUACCUGCAAGAGCUCGGGUUGAGUGUUAUAGUGGCUUUAAACCAAAUAACACUAGUGCACAAUACAUAUUUAGCACUGUUAACGCAUUACCUGAGUGUAAUGCUGAUAUCGUUGUUGUAGAUGAAGUUUCAAUGUGUAC
AAAUUAUGACCUUUCUGUUAUUAAUCAGCGUUUAUCAUAUAAACAUAUUGUUUAUGUUGGUGAUCCACAACAACUUCCUGCACCUAGAGUAAUGAUUACUAAAGGUGUUAUGGAGCCUGUUGAUUAUAACGUUGUUACUCAACGUAUGUGUGCUAUAGGCCCUGAUGUUUUUCUUCAUAAAUGUU
AUAGAUGUCCUGCUGAAAUAGUUAAUACAGUUUCUGAACUGUUUUAUGAGAACAAGUUUGUCCCUGUUAAACCUGCUAGUAAACAGUGUUUUAAAGUCUUUUUAAGGGUAAUGUACAGGUUGACAAUGGUUCUAGUAUUAACAGAAAGCAGCUUGAAAUAGUUAAGCUGUUUUUAGUUAAAAAU
CCAAGUUGGAGUAAGGCUGUGUUUAUUUCUCCUUAUAAUAGUCAGAAUUAUGUUGCUAGUAGAUUUUUAGGACUUCAAAUUCAAACUGUUGAUUCUUCUCAAGGUAGUGAGUAUGAUUAUGUAAUCUAUGCACAAACUUCUGACACUGCACAUGCUUGCAAUGUAAACCGUUUUAAUGUUGCUAU
AACACGUGCUAAGAAGGGUAUAUUUUGUGUAAUGUGUGAUAAAACUUUGUUUGAUUCACUUAAGUUUUUUGAGAUUAAACAUGCAGAUUUACACUCUAGCCAGGUUUGUGGCUUGUUUAAAAAUUGUACACGCACUCCUCUUAAUUUACCACCAACUCAUGCACACACUUUCUUGUCGUUGUCAG
AUCAGUUUAAGACUACAGGUGAUUUAGCUGUUCAAAUAGGUUCAAAUAACGUUUGUACUUAUGAACAUGUUAUAUCAUUUAUGGGUUUUAGGUUUGAUAUUAGUAUUCCUGGUAGUCAUAGUUUGUUUUGUACACGUGACUUUGCUAUUCGCAAUGUGCGUGGUUGGUUGGGUAUGGAUGUUGAA
AGUGCUCAUGUUUGUGGCGAUAACAUAGGUACUAAUGUUCCUUUACAGGUUGGUUUUUCAAAUGGGUGUUAAUUUUGUUGUGCAAACUGAAGGUUGUGUGUCUACCAAUUUUGGUGAUGUUAUUAAACCUGUUUGUGCAAAAUCUCCACCAGGUGAACAAUUUAGACACCUUGUUCCUCUUUUACG
UAAAGGACAACCUUGGUUAAUUGUCCGUAGACGCAUUGUGCAAAUGAUACUGAUUAUUUGUCCAAUUUGUCUGACAUUCUUGUCUUUGUUUUGUGGGCAGGUAGUUUGGAAUUAACUACAAUGCGUUACUUUGUAAAAAUAGGGCCAAUUAAAUAUUGUUAUUGUGGUAAUUCUGCCACUUGU
AUAAUUCAGUUAGUAAUGAAUAUUGUUGUUUUAAACAUGCAUUGGGUUGUGAUUAUGUUUACAAUCCGUAUGCUUUUGAUAUACAACAGUGGGGUUAUGUUGGUUCCUUGAGCCAAAACCACCACACAUUCUGUAACAUUCAUAGAAACGAGCAUGAUGCCUCUGGUGAUGCUGUUAUGACACGU
UGUUUGGCAGUACAUGAUUGUUUUUGUCAAAAAUGUUGAUUGGACUGUAACGUACCCCUUUAUUGCAAAUGAGAAAUUUAUCAAUGGCUGUGGGCGUAAUGUCCAGGGACAUGUGUUCGUGCAGCCUUGAAAUUGUAUAAACCUAGUGUUAUUCAUGACAUUGGUAAUCCUAAAGGUGUACGUUG
UGCUGUUACUGAUGCCAAAUGGUACUGUUAUGACAAGCAACCUGUUAAUAGUAAUGUCAAGUUGUUGGAUUAUGAUUAUGCAACUCAUGGUCAACUUGAUGGUCUUUGUCUAUUCUGGAAUUGUAAUGUUGAUAUGUAUCCAGAAUUUUCAAUUGUGUGUCGUUUGACACACGUACUCGUUCUG
UUUUUAAUUUAGAAGGUGUUAAUGGUGGUUCACUCUUUAUGUUAACAAACAUGCGUUUCAUACACCAGCAUAUGAUAAACGUGCUUUUGUUAAAUUAAAACCUAUGCCCUUUUUUUACUUUGAUGACAGUGAUUGUGAUGUGGUGCAAGAACAAGUAAUUAUGUACCCCUUCGCGUAGUAGUUG
GUUACCCGUUGUAAUAUAGGUGGUGCUGUUUGUUCAAAACAUGCAAAUUGUAUCAAAAAUAUGUUGAGGCAUAUAAUACAUUUACACAGGCUGGUUUUAACAUUUGGGUACCACAUAGUUUUGAUGUUUAUAAUUUGUGGCAAAUUUUUAUUGAAACUAAUUUACAAAGUCUUGAAAAUAUAGC
AUUUAAUGUUGUAAAAAAAGGGUGUUUUACUGGUGUUGAUGGUGAGUUACCUGUUGCAGUUGUUAACGACAAAGUUUUUGUUCGCUAUGGCGAUGUUGACAACUUAGUUUUUACAAAUAAAACAACAUUGCCUACUAAUGUUGCUUUUGAAUUGUUUGCAAAACGAAAAAUGGGUUUAACACCAC
CAUUGUCUAUUCUCAAAAAUCUCGGUGUUGUUGCUACAUAUAAAUUUGUUUUAUGGGAUUAUGAAGCUGAAAGACCUUUUACCUCAUAUACUAAGAGUGUAUGUAAAUACACUGAUUUUAAUGAGGAUGUUUGUGUUUGUUUUGACAAUAGUAUUCAGGGUUCGUAUGAGCGUUUUACGCUUACU
ACGAACGCUGUUUUAUUUUCUACUGUUGUCAUUAAAAAUUUAACACCUAUAAAGUUGAAUUUUGGUAUGUUGAAUGGUAUGCCAGUUUCUUCUAUUAAGGGUGAUAAAGGUGUUGAAAAAUUAGUUAAUUGGUACAUAUAUGUUCGUAAAAAUGGUCAAUUUCAAGAUCACUAUGAUGGUUUUUA
CACUCAAGGUAGGAAUUUAUCAGACUUUACACCAAGAAGUGAUAUGGAGUAUGAUUUUCUUAACAUGGAUAUGGUGUUGUUUUUAUUAAAUAAUGGUCUUGAGGAUUUUAAUUUUGAACAUGUGUAUAUGGUGAUGUUUCAAAAACUACAUUAUAGGAGGUCUUCAUUUGUUGAUAUCACAGUUUA
GGCUUAGUAAAAUGGGUGUUUUGAAAGCUGAUGAUUUUGUCAUUGCUUCUGACACAACUUUGAGGUGCUGCACUGUUACUUAUCUUAAUGAACUUAGUUCAAAAGUUGUUUGUACUUAUAUGGAUUUGUUGUUGGACGACUUUGUUACUAUACUAAAGAGUUUAGAUCUUGGUGUAAUAUCUAAA
GUUCAUGAAGUUAUUAUAGAUAAUAAACCUUAUAGGUGGAUGUUGUGGUGUAAAGAUAACCACUUGUCCACUUUUUAUCCACAGUUGCAGUCUGCUGAAUGGAAGUGUGGUUAUGCUAUGCCACAAAUUUAUAAGCUUCAACGUAUGUGUUUGGAACCUUGUAAUUUAUAUAAUUAUGGUGCUGG
UAUUAAGUUGCCUAGUGGUAUAAUGUUAAAUGUUGUUAAAUACACUCAGCUUUGUCAAUACCUAAAUAGCACUACAAUGUGCGUACCUCAUAAUAUGCGUGUUUUGCACUAUGGUGCUGGUUCUGACAAAGGUGUGGCACCUGGUACAACUGUUUUAAAACGUUGGCUACCACCUGAUGCAAUAA
UCAUUGAUAAUGACAUCAAUGAUUAUGUUAGUGAUGCAGAUUUUAGCAUUACAGGUGAUUGUGCUACUGUUUACCUUGAAGAUAAGUUUGACUUACUUAUUUCUGAUAUGUAUGAUGGUAGAAUUAAAUUUUUACUUAUCUUAAUGGUUU
AUUAGAGAAAAAUUAGCUAUUGGUGGUAGUGUUGCCAUUAAGAUUACAGAAUAUAGUUGGAAUAAGUAUCUUUAUGAAUUAAUACAGAGAUUUGCUUUUUGGACUUUGUUUUGCACGUCUGUUAAUACAUCCUCUUCAGAAGCUUUUCUUAUUGGUAUUAAUUAUUUAGGUGACUUUAUUCAAGG
UCCUUUUAUAGCUGGUAACACUGUUCAUGCUAAUUAUAUAUUUUGGCGUAAUUCUACUAUUAUGUCUUUGUCAUACAAUUCAGUUUUAGAUCUAAGUAAGUUUGAAUGUAAACAUAAGGCCACUGUUGUUGUUACACUUAAAGAUAGUGAUGUAAAUGAUAUGGUUUUGAGUUUGAUUAAGAGUG
GUAGGUUGUUGUUACGUAAUAGUGGCCGUUUUGGUGGUUUUAGUAAUCAUUUAGUCUCAACUAAAUGAAACUUUUCUUGAUUUUGCUUAUUUUGCCCCUGGUUUCUUGCUUUUCUACAUGUAACAGUAAUGCUAGUAUUUCUAUGUUACAAUUAGGUGUUCCUGAUAACUCUUCAACUAUUGUCA
CAGGUUUGUUGCCAGUCCAUUGGAUUUGUGCUAAUCAGAGCACAUCUAGUUACCCAGCCAAUGGCUUUUUCUAUAUUGAUGUCGGUAAACACCGUAGUGCUUUUGCACUCCAUAGUGGUUAUUAUGAUGCUAACCAGUAUUAUAUUUAUCUCACUAAUAAAAUACAUUUAAAUGCUCCUGUCACU
CUGAAGAUUUGUAAGUUUGGAAACACUUCUUUUGAUUUUUAAGUAAUGUUUCUACUUCUCAUCAUUGUAAUAGUUAAUUCGUCAUUCACAGAACAGUUAGGUGUGCCUUUGGGCAUAACUAUCGGGUGAAACUGUACGUUUGCAUUUAUAUAAUGCAACUCGCACCUUUUAUGUGCCUGCCGC
UUAUAAACUUACUAAACUUAGUGUUAAAUGUUACUUUAGUGAAUCCUGUGUUUUUAGUGUUGUCAAUGCUACCGUUACUGUUAAUGUCACCACACUUAAUGGCCGUAUAGUUAAUUACACUGUUUGUGAUGAUUGUAAUGGUUACACUGAUAACAUAUUUUCUGUUCAACAGGAUGGCCGCAUUC
CUAAUGGUUUUUCUUUUAAUAAUUGGUUUUUGUUAACUAAUGGUUCUACACUAGUGGACGGUGUCUCUAGACUUUACCAACCACUUCGUUUAACUUGUUUUAUGGCCUGUACCUGGUCUUAAAUCUUCGACUGGUUUUUGUUUAUUUUAACGCCACUGGUUCUGAUGUUAAUUGUAAUGGUUAUCAA
CAUCAUUCUGUUGCUGAUGUUAUGCGUUACAAUCUUAACUUCAGUUCUAAUUCUGUGGAUAAUCUCAAGAGUGGUGUUAUAGUUUUUAAAACUCUACAGUACGAUGUUUUGUUUUACUGUAGUAAUUCUUCUUCGGGCGUUCUAGACACCACAAUACCUUUUGGCCCUUCCUCUCAACCUUAUUA
CUGUUUUAUAAAUAGUACCAUCAACACCACUCAUGUUAGCACUUUUGUGGGUGUCCUACCACCCAUUGUGCGUGAGAUUGUUGUGCUAGAACUGGUCAGUUUUAUAUUAAUGGUUUUAAGUAUUUUGAUUUGGGUUUUAUCGAAGCUGUUAAUUUUAAUGUCACGACAGCUAGUGCUACAGAUU
UUUGGACGGUUGCAUUUGCUACUUUUGUUGAUGUUUUGGUUAAUGUUAGUGCAACUAACAUUCAAAACUUACUUUAUUGCGAUUCUCCAUUUGAAAAGUUGCAGUGUGAGCACUUGCAGUUUGGAUUGCAAGAUGGUUUUUAUUCUGCAAAUUUCUUGAUGAUAAUGUUUUGCCGUAUACUUAU
GUUGCACUUCCUAUUUAUUAUCAACAUACUGAUAUAAAUUUUACUGCGACUGCAUCUUUUGGUGGUUCUUGUUAUGUUUGCAAACCACAUCAGGUUAAUAUAUCUCUUAAUGGUAAUACUUCAGUGUGUGUUAGAACAUCUCAUUUUUCAAUUAGGUAUAUUUAUAACCGCGUUAAGAGUGGUUC
ACCAGGUGACUCUUCAUGGCACAUUUAUUUAAAGAGUGGCACUUGUCCAUUUUCUUUUUCUAAGUUAAAUAAUUUUCAAAAGUUUAAGACUAUUUGUUUCUCAACCGUCGUAGUGCCUGGUAGUUGUAAUUUUCCACUUGAAGCCACCUGGCAUUACACUUCUUAUACUAUGUUGGUGCUUUGU
AUGUUACUUGGUCUGAAGGUAAUUCCAUUACUGGUGUACCUUAUCCUGUCUCUGGUAUUCGUGAGUUUAGUAAUUUAGUUUUAAAUAAUUGUACCAAAUAUAAUAUUUAUGAUUAUGUUGGUACUGGAAUUAUACGUUCUUCAAACCAGUCACUUGCUGGUGGUAUUACAUAUGUUUCUAACUCU
GGUAAUUUACUUGGUUUUAAAAUGUUUCCACUGGUAACAUUUUUAUUGUGACACCAUGUAACCAACCAGACCAGAGUAGCUGUUUAUCAACAAAGCAUUAUUGGUGCCAUGACUGCUGUUAAUGAGUCUAGAUAUGGCUUGCAAAACUUACUACAGUUACCUAACUUUUAUUAUGUUAGUAAUGG
UGGUAACAAUUGCACUACGGCCGUUAUGACUUAUUCUAAUUUUGGUAUUUGUGCUGAUGGUUCUUUAAUUCCUGUUCGUCCGCGUAAUUCUAGUGAUAAUGGUAUUUCAGCCAUAGUCACUGCUAAUUUAUUCCAUUCCCUCUAACUGGACUACUUCAGUUCAAGUUGAGUACCUCCAAAUUACUA
GUACUCCAAUAGUUGUUGAUUGUGCUACUUAUGUGUGUAAUGGUAACCCUCGCUGUAAGAAUCUACUUAAGCAGUAUACUUCUGCUUGUAAAACUAUUGAAGAUGCCUUACGACUUAGUGCUCAUUUGGAAACUAAUGAUGUUAGUAGUAUGCUAACUUUUGAUAGCAAUGCUUUUAGUUUGGCU
AAUGUUACUAGUUUUGGAGAUUAUAACCUUUCUAGUGUUUUACCUCAGAGAAACAUUCAUUCAAGCCGUAUAGCAGGACGUAGUGCUUUGGAAGAUUUGUUGUUUAGCAAAGUUGUUACAUCUGGUUUGGGUACUGUUGAUGUUGACUAUAAGUCUUGUACUAAAGGUCUUUCUAUUGCUGACCU
UGCUUGUGCUCAGUACUACAAUGGCAUAAUGGUUUUGCCAGGUGUUGCUGAUGCUGAACGUAUGGCCAUGUACACAGGUUCUCUUAUAGGUGGCAUGGUGCUCGGAGGUCUUACAUCAGCAGCCGCCAUACCUUUUUCUUUGGCACUGCAAGCACGACUUAACUAUGUUGCUUUACAAACUGAUG

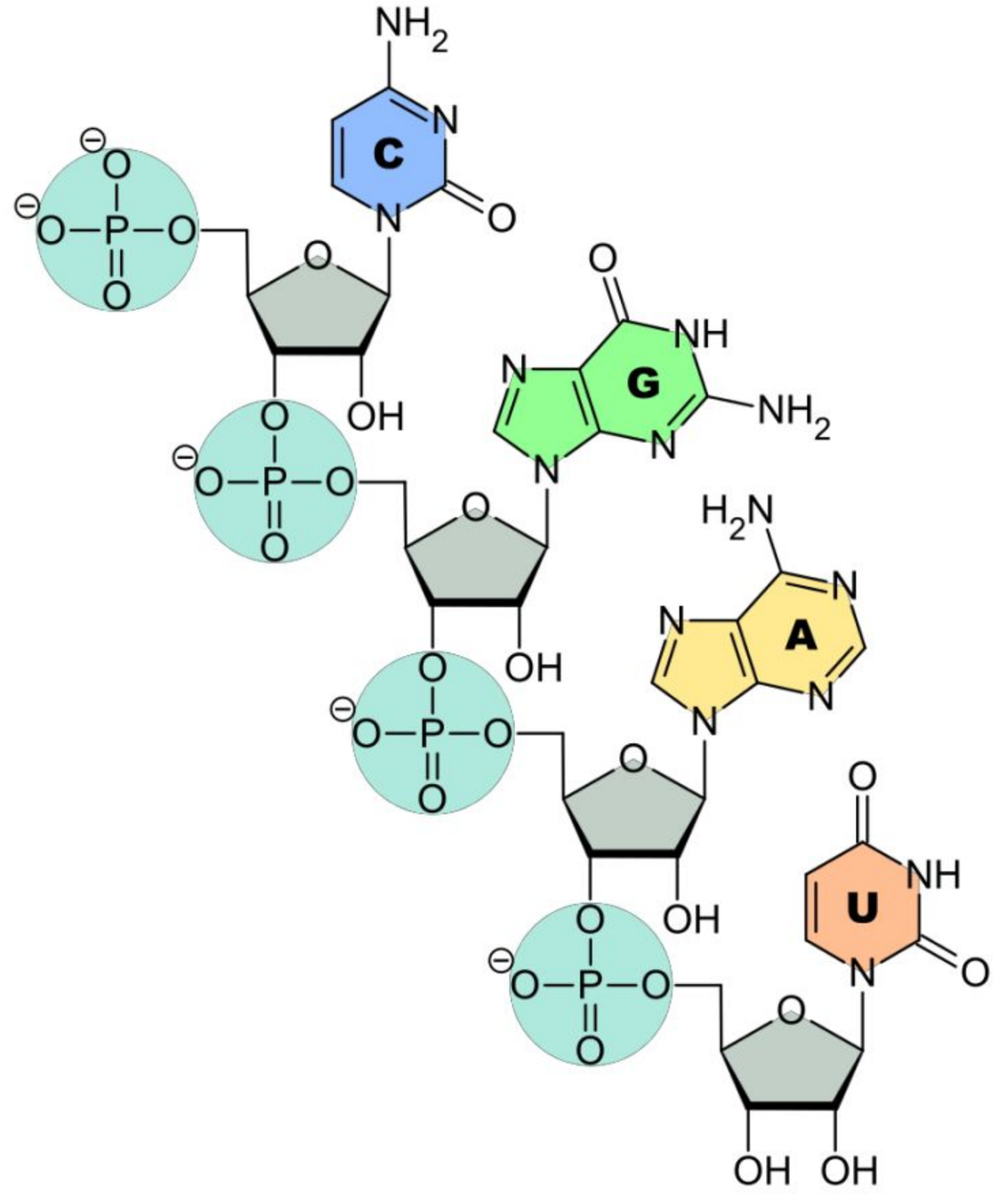
```
UGCUUCAAGAAAAUCAGAAAAUUUUGGCUGCAUCAUUUAAUAAGGCUAUUAAUAACAUUGUUGCUUCUUUUAGUAGCGUUAAUGAUGCUAUUACACAAACUGCAGAGGCUAUACAUACUGUUACUAUUGCACUUAAUAAGAUUCAGGAUGUUGUUAAUCAACAGGGUAGUGCUCUUAACCAUCUC
ACUUCACAAUUGAGACAUAAUUUUCAGGCCAUUUCUAAUUCAAUUCAAGCUAUUUAUGACCGGCUUGAUUCAAUUCAAGCCGAUCAACAAGUUGACAGAUUAAUUACUGGACGGCUUGCAGCUUUGAAUGCAUUUGUUUCCCAAGUUUUGAAUAAAUAUACUGAAGUUCGUAGUUCCAGACGCUU
AGCACAGCAGAAGAUUAAUGAAUGUGUCAAAUCACAAUCUAAUAGAUAUGGUUUUUGUGGCAAUGGCACCCAUAUCUUUUCAAUCGUCAACUCAGCUCCAGAUGGUUUGCUUUUUCUUCAUACUGUUUUGCUGCCAACCGACUACAAGAAUGUAAAGGCGUGGUCGGGUAUCUGUGUUUGAUGGCA
UUUACGGCUAUGUUCUGCGUCAACCUAACUUGGUUCUUUAUUCUGAUAACGGUGUCUUUCGUGUAACUUCCAGGGUCAUGUUUCAACCUCGUUUACCUGUUUUGUCUGAUUUUGUGCAAAUAUAUAAUUGUAAUGUUACUUUUGUUAACAUAUCUCGUGUCGAGUUACAUACUGUCAUACCUGAC
UACGUUGAUGUUAAUAAAACAUUACAAGAGUUUGCACAAAACUUACCAAAGUAUGUUAAGCCUAAUUUUGACUUGACUCCUUUUAAUUUAACAUAUCUUAAUUUGAGUUCUGAGUUGAAGCAACUCGAAGCUAAAACUGCUAGUCUUUUCCAAACUACUGUUGAAUUACAAGGUCUUAUUGAUCA
GAUUAACAGUACAUAUGUUGAUUUGAAGUUGCUUAAUAGGUUUGAAAAUUAUAUUAAAUGGCCUUGGUGGGUUUGGCUCAUUAUUUCUGUUGUUUUUGUUGUAUUGUUGAGUCUUCUUGUGUUUUGUUGUCUUUCUACAGGUUGUUGUGGUUGUUGCAAUUGUUUAACUUCAUCAAUGCGAGGUU
GUUGUGAUUGUGGUUCAACUAAACUUCCUUAUUACGAAUUUGAAAAGGUCCACGUUCAAUAAUGCCUUUCGGUGGCCUAUUUCAACUUACUCUUGAAAGUACUAUUAAUAAGAGUGUGGCUAAUCUCAAAUUACCACCUCAUGAUGUUACUGUCUUGCGUGACAAUCUCAAACCUGUUACUACAC
UUAGUACUAUCACUGCUUAUUUGUUAGUUAGUUUGUUUGUCACUUAUUUUGCUUUAUUCAAACCUCUUACUGCUAGAGGUCGUGUUGCUUGUUUUGUUUUAAAACUAUUGACACUAUUUGUCUAUGUGCCUUUAUUGGUUCUUUUUGGUAUGUAUCUUGACAGUUUUAUAAUUUUUUCUACGCUG
UUGUUUCGAUUCAUACAUGUUGGCUAUUAUGCCUAUCUCUAUAAAAAUUUUUCAUUUGUUUUGUUCAAUGUUACUAAACUAUGCUUCGUUUCAGGCAAGUGUUGGUAUCUUGAACAAUCAUUUUAUGAAAAUCGUUUUGCUGCUAUUUAUGGUGGUGACCACUAUGUCGUUUUAGGUGGUGAAAC
UAUUACUUUUGUUUCUUUUGAUGACCUUUAUGUUGCUAUUAGAGGUUCUUGUGAAAAGAACCUACAACUUAUGCGUAAGGUUGACUUGUAUAAUGGUGCUGUCAUUUACAUUUUUGCCGAAGAGCCUGUUGUUGGUAUGUCUACUCCUCUCAACUAUACGAAGAUGUUCCUUCGAUUAAUUGAU
GACAAUGGCAUUGUCCUCAAUUCCAUUUUAUGGCUCCUUGUUAUGAUAUUUUUCUUUGUGUUGGCAAUGACUUUUAUUAAACUGAUUCAAUUGUGUUUUACUUGUCAUUAUUUUUUUAGUAGGACAUUAUAUCAACCAGUUUAUAAAAUUUUUCUUGCUUACCAAGAUUAUAUGCAAAUAGCACC
UGUUCCAGCUGAAGUACUAAAUGUCUAAACUAAACGAUGUCUAAUAGUAGUGUGCCUCUUUCAGAGGUUUAUGUCCAUUUACGUAACUGGAACUUUAGUUGGAAUUUAAUUCUAACGCUUUUUAUAGUUGUGUUGCAGUAUGGGCAUUAUAAGUAUAGCAGACUUCUUUAUGGUUUAAAGAUGUC
UGUUUUAUGGUGUUUAUGGCCACUUGUUCUAGCUUUGUCUAUUUUUGACUGUUUUGUCAAUUUUAAUGUGGACUGGGUCUUUUUUGGUUUUAGUAUUCUUAUGUCUAUUAUUACACUUUGUUUGUGGGUUAUGUAUUUUGUUAAUAGUUUCAGACUUUGGCGCCGUGUUAAAACUUUUUGGGCUU
UUAAUCCUGAAACUAAUGCAAUCAUCUCUCUCCAGGUUUACGGACAUAAUUAUUACUUACCGGUGAUGGCUGCACCUACAGGUGUUACAUUAACACUUCUCAGUGGUGUACUUCUUGUUGAUGGCCAUAAGAUUGCUACUCGUGUUCAAGUGGGUCAGUUGCCUAAAUAUGUAAUAGUUGCUACA
CCUAGUACCACAAUUGUUUGUGACCGUGUUGGUCGCUCUGUUAAUGAAACAAGCCAGACUGGUUGGGCAUUCUACGUCCGUGCUAAACAUGGUGAUUUUUCUGGUGUUGCCUCUCAGGAGGGUGUUUUGUCAGAAAGAGAGAAGUUGCUUCAUUUAAUCUAAACUAAACAAAAUGGCUAGUGUAA
AUUGGGCCGAUGACAGAGCUGCUAGGAAGAAAUUUCCUCCUCCUUCAUUUUACAUGCCUCUUUUGGUUAGUUCUGAUAAGGCACCAUAUAGGGUCAUCCCCAGGAAUCUUGUCCCUAUUGGUAAGGGUAAUAAAGAUGAGCAGAUUGGUUAUUGGAAUGUUCAAGAGCGUUGGCGUAUGCGCAGG
GGGCAACGUGUUGAUUUGCCUCCUAAAGUUCAUUUUUAUUACCUAGGUACUGGACCUCAUAAGGACCUUAAAUUCAGACAACGUUCUGAUGGUGUUGUUUGGGUUGCUAAGGAAGGUGCUAAAACUGUUAAUACCAGUCUUGGUAAUCGCAAACGUAAUCAGAAACCUUUGGAACCAAAGUUCUC
UAUUGCUUUGCCUCCAGAGCUCUCUGUUGUUGAGUUUGAGGAUCGCUCUAAUAACUCAUCUCGUGCUAGCAGUCGUUCUUCAACUCGUAACAACUCACGAGACUCUUCUCGUAGCACUUCAAGACAACAGUCUCGCACUCGUUCUGAUUCUAACCAGUCUUCUUCAGACCUUGUUGCUGCUGUUA
CUUUGGCUUUAAAGAACUUAGGUUUUGAUAACCAGUCGAAGUCACCUAGUUCUUCUGGUACUUCCACUCCUAAGAAACCUAAUAAGCCUCUUUCUCAACCCAGGGCUGAUAAGCCUUCUCAGUUGAAGAAACCUCGUUGGAAGCGUGUUCCUACCAGAGAGGAAAAUGUUAUUCAGUGCUUUUGU
CCUCGUGAUUUUAAUCACAAUAUGGGGGAUUCAGAUCUUGUUCAGAAUGGUGUUGAUGCCAAAGGUUUUCCACAGCUUGCUGAAUUGAUUCCCAAUCAGGCUGCGUUAUUCUUUGAUAGUGAGGUUAGCACUGAUGAAGUGGGUGAUAAUGUUCAGAUUACCUAUACCUACAAAAUGCUUGUAGC
UAAGGAUAAUAAGAACCUUCCUAAGUUCAUUGAGCAGAUUAGUGCUUUUACUAAACCCAGUUCUAUCAAAGAAAUGCAGUCACAAUCAUCUCAUGCUGUUCAGAACACAGUACUUAAUGCUUCUAUUCCAGAAUCUAAACCAUUGGCUGAUGAUGAUUCAGCCAUUAUAGAAAUUGUCAACGAGG
UUUUGCAUUAAAUUGUUUUGUAAUUCCAGUUGAAUGUUUAUUAUUAUUAGUUGCAACCCCAUGCGUUUAGCGCAUGAUAAGGGUUUAGUCUUACACACAAUGGUAGGCCAGUGAUAGUAAAGUGUAAGUAAUUUGCUAUCAUAUUAACAUGUCUAGAGGAAAGUCAGAACUUUUUCUGUUUGUGU
UGUUGGAGUACUUAAAGAUCGCAUAGGCGCGCCAACAAUGGAAGAGCCAACAACAUAUCUAAAAAUGUUUUGUCUGGUACUUGUUAAUGAUA'
```

The RNA genome is made of phosphate groups, ribose sugars, and four nucliotides, which are cytosine, guanine, adonine, and uricil.

In [32]:
```
Image(filename='RNA_molecule.jpg')
```

Out[32]:

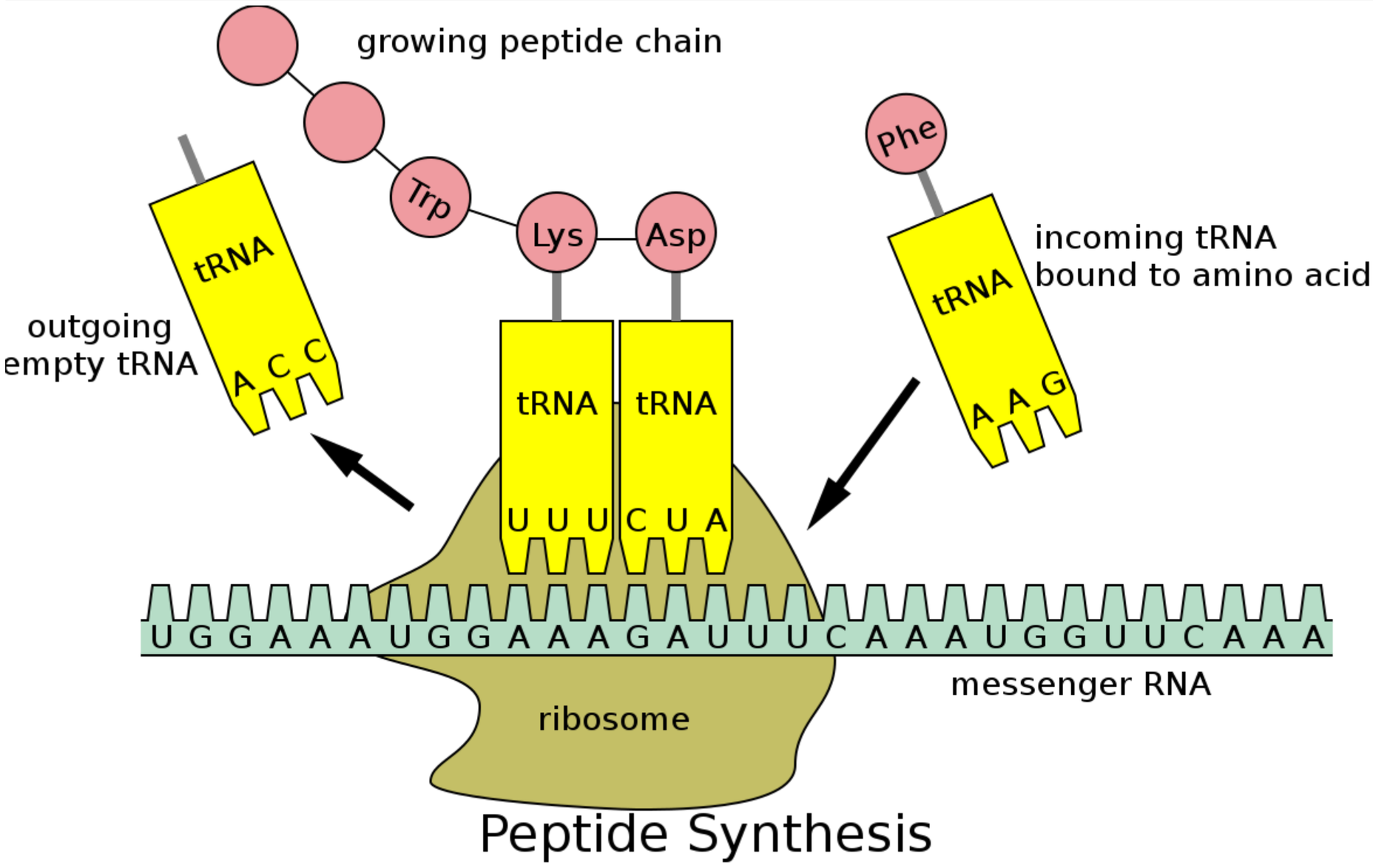
RNA



The RNA is translated to proteins in three steps: initiation, which is when a ribosome, mRNA (sgRNA is a type of mRNA), and tRNA connect; elongation, which is when tRNA transfers codons to anticodons, and then brings amino acids to the ribosome to be linked; and termination, which is when the polypeptide chain is complete.

In [225…
```
Image(filename='translation.png')
```

Out[225…



Data Mining

In the genome is an '\n' character that is python for new line that will be removed.

In [38]:
```
set(genome)
```

```
{'\n', 'A', 'C', 'G', 'U'}
```

Out[38]:

Data Cleaning

Remove '\n' from genome and protein strings.

In [53]:
```
genome=genome.replace('\n','')
```

Makes a dataframe of sars-cov-2 mRNA nucleotides and mRNA antinucleotides.

In [54]:
```
nucleotides_df=pd.DataFrame(columns=['Nucleotides'])
for i in genome:
    nucleotides_df.loc[len(nucleotides_df.index)] = [i]
nucleotides_df
```

Out[54]:

	Nucleotides
0	A
1	C
2	U
3	U
4	U
...	...
27464	U
27465	G
27466	A
27467	U
27468	A

27469 rows × 1 columns

Adds a lagged RNA nucleotides column for nucleotide sequence prediction.

In [55]:
```
nucleotides_df['Nucleotides_lagged']=nucleotides_df['Nucleotides'].shift(-1)
nucleotides_df
```

Out[55]:

	Nucleotides	Nucleotides_lagged
0	A	C
1	C	U
2	U	U
3	U	U
4	U	G
...	...	...
27464	U	G
27465	G	A

	Nucleotides	Nucleotides_lagged
27466	A	U
27467	U	A
27468	A	NaN

27469 rows × 2 columns

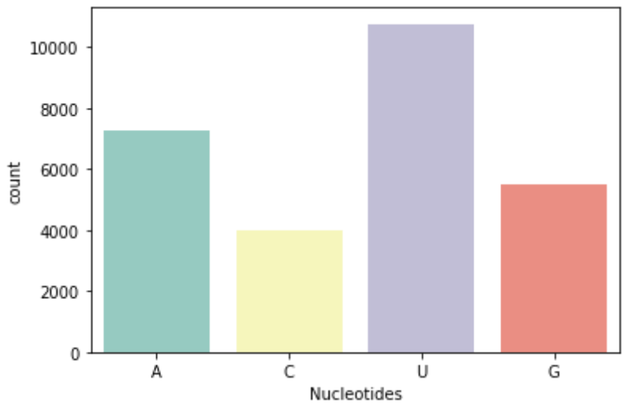
In [9]:
```
nucleotides_df['Nucleotides_lagged']=nucleotides_df['Nucleotides_lagged'].fillna('A')
```

Data Exploration

Sars-cov-2 genome is made up of apoximatly 30% adenine, 18% guanine, 20% cytosine and 32% uracil.

In [894…
```
sns.countplot(nucleotides_df['Nucleotides'], palette='Set3')
```

Out[894…
```
<AxesSubplot:xlabel='Nucleotides', ylabel='count'>
```



Percentage of uricil in genome data.

In [901…
```
len(nucleotides_df['Nucleotides'].loc[nucleotides_df['Nucleotides']=='U'])/len(nucleotides_df['Nucleotides'])*100
```

Out[901…
```
39.14594633950999
```

Percentage of guanine in genome data.

In [902…
```
len(nucleotides_df['Nucleotides'].loc[nucleotides_df['Nucleotides']=='G'])/len(nucleotides_df['Nucleotides'])*100
```

Out[902…
```
20.01528996323128
```

Percentage of adonine in genome data.

In [903…
```
len(nucleotides_df['Nucleotides'].loc[nucleotides_df['Nucleotides']=='A'])/len(nucleotides_df['Nucleotides'])*100
```

Out[903…
```
26.338781899595908
```

Percentage of cytosine in genome data.

In [904…
```
len(nucleotides_df['Nucleotides'].loc[nucleotides_df['Nucleotides']=='C'])/len(nucleotides_df['Nucleotides'])*100
```

Out[904…
```
14.49998179766282
```

Feature Engineering

Encodes nucleotides to numbers.

In [46]:
```
nucleotides_df['Nucleotides']=nucleotides_df['Nucleotides'].astype('category')
nucleotides_df['Nucleotides']=nucleotides_df['Nucleotides'].cat.codes
nucleotides_df['Nucleotides_lagged']=nucleotides_df['Nucleotides_lagged'].astype('category')
nucleotides_df['Nucleotides_lagged']=nucleotides_df['Nucleotides_lagged'].cat.codes
```

Makes independent and dependent variable.

In [47]:
```
X=nucleotides_df['Nucleotides']
y=nucleotides_df['Nucleotides_lagged']
```

Makes train, validation, and test set.

In [48]:
```
X_train= X[:21000]
y_train= y[:21000]
X_val=X[21001:21601]
y_val=y[21001:21601]
X_test=X[21601:24601]
y_test=y[21601:24601]
```

Encodes numbers to lists of zeros and a one that represents the class.

In [10]:
```
X_train = to_categorical(X_train)
y_train = to_categorical(y_train)
X_val = to_categorical(X_val)
y_val = to_categorical(y_val)
X_test = to_categorical(X_test)
y_test = to_categorical(y_test)
```

Splits data into threes to represent each codon.

In [11]:
```
X_train = [X_train[x:x+3] for x in range(0, len(X_train),3)]
y_train=[y_train[x:x+3] for x in range(0, len(y_train),3)]
X_val = [X_val[x:x+3] for x in range(0, len(X_val),3)]
y_val=[y_val[x:x+3] for x in range(0, len(y_val),3)]
X_test = [X_test[x:x+3] for x in range(0, len(X_test),3)]
y_test=[y_test[x:x+3] for x in range(0, len(y_test),3)]
```

Makes data into arrays.

In [12]:
```
X_train=np.array(X_train)
y_train=np.array(y_train)
X_val=np.array(X_val)
y_val=np.array(y_val)
X_test=np.array(X_test)
y_test=np.array(y_test)
```

The model will train on 7000 codons in order to predict the next nucleotide.

In [13]:
```
X_train.shape
```

Out[13]: (7000, 3, 4)

In [14]:
```
y_train.shape
```

Out[14]: (7000, 3, 4)

In [15]:
```
X_train
```

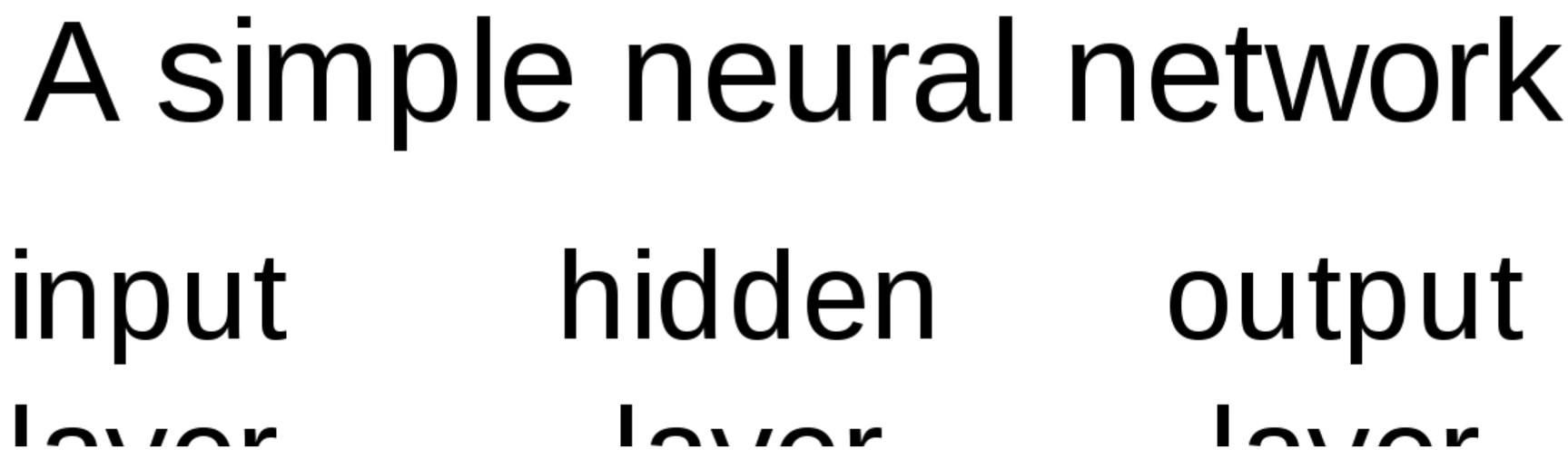
Out[15]:
```
array([[[1., 0., 0., 0.],
        [0., 1., 0., 0.],
        [0., 0., 0., 1.]],

       [[0., 0., 0., 1.],
        [0., 0., 0., 1.],
        [0., 0., 1., 0.]],

       [[0., 0., 0., 1.],
        [0., 0., 1., 0.],
        [0., 0., 0., 1.]],

       ...,

       [[0., 1., 0., 0.],
        [0., 1., 0., 0.],
        [1., 0., 0., 0.]],

       [[0., 1., 0., 0.],
        [1., 0., 0., 0.],
        [0., 1., 0., 0.]],

       [[0., 0., 0., 1.],
        [0., 0., 0., 1.],
        [1., 0., 0., 0.]]], dtype=float32)
```

RNN Model

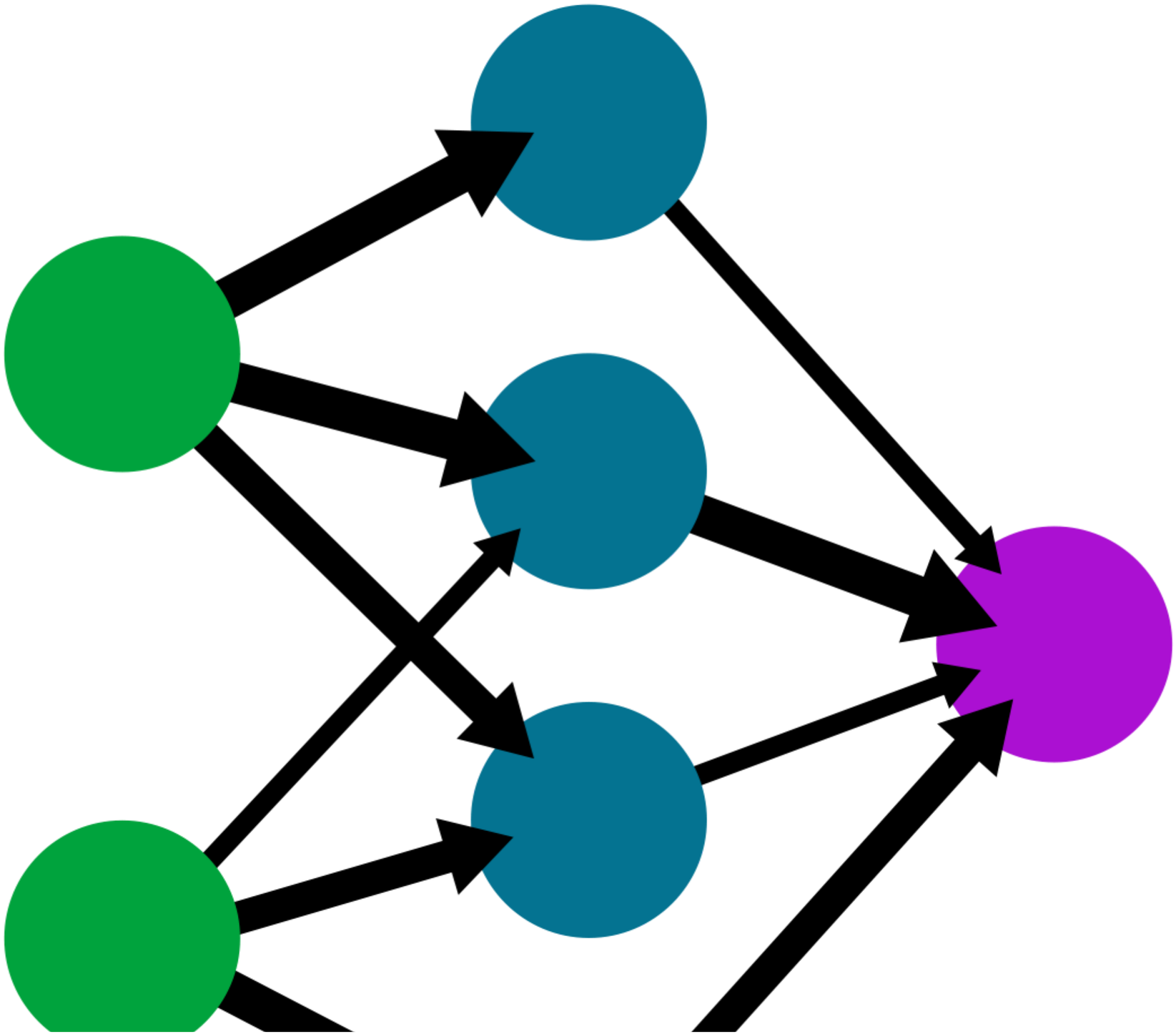
A simple neural network has an input layer followed by a hidden layer and an output layer.

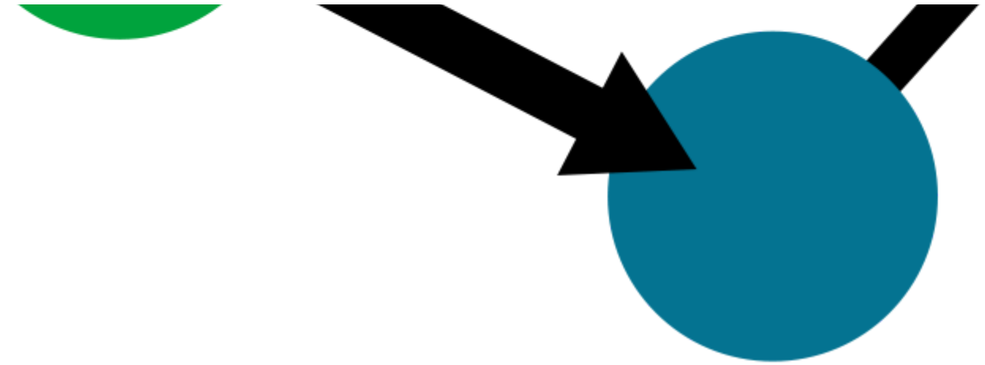
In [38]:
```
Image(filename='ANN.png')
```

Out[38]:



layer layer layer

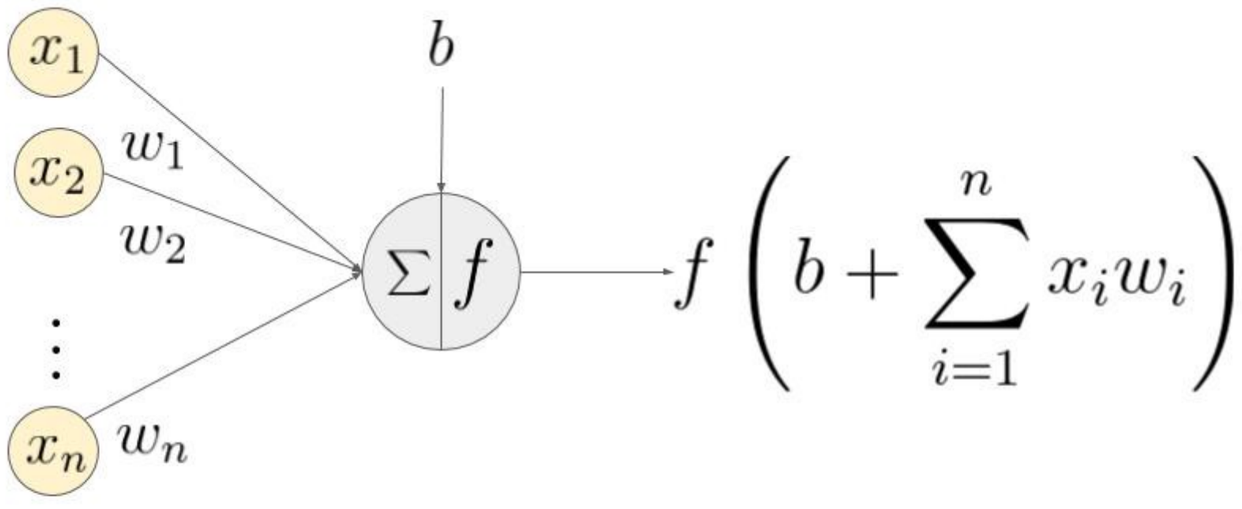




A neural network uses weights and bias through an activation function to determine which neurons will be passed to the next layer of the network.

In [39]:
```
Image(filename='wb.jpeg')
```

Out[39]:



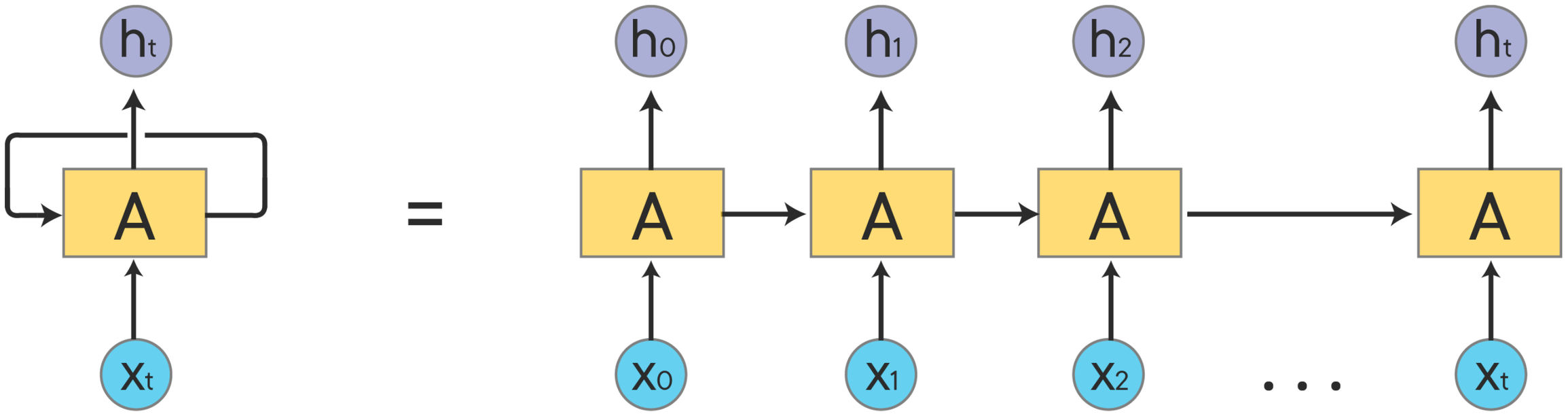
An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

A Recurrent Neural Network is a neural network that passes its output, h, from a layer back into itself as input for the next layer. The model starts at the most recent output, and then works backward to calculate the loss and update the weights at each time step. The hidden state, ht, is a function of the current input, xt, the previous hidden state, ht-1, and the bias, bt.

$$h_t = \sigma(W * x_t + U * h_{t-1} + b_t)$$

In [33]:
```
Image(filename='RNN.png')
```

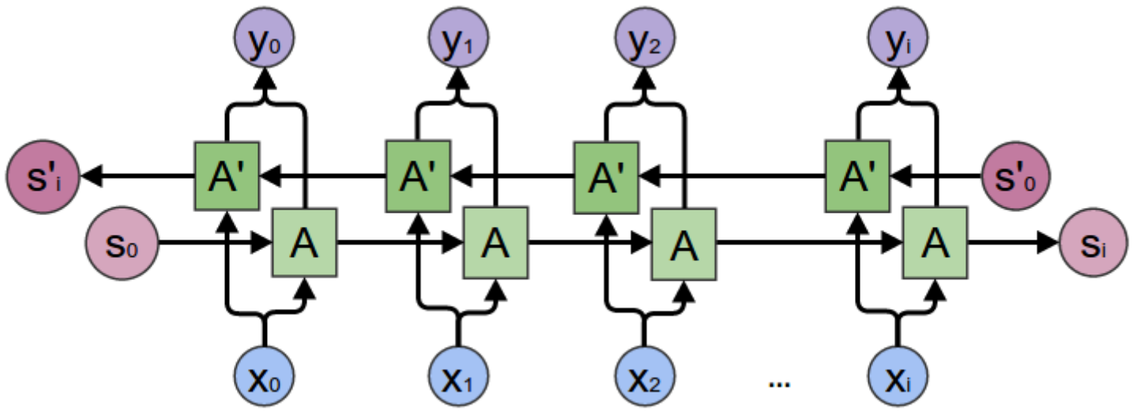
Out[33]:



In a bidirectional RNN, the input sequence is passed forward in normal time order through one network, and then in reverse time order through another network. The outputs of the two networks are then concatenated at each time step. This enables the resulting network to have backward and forward information about the sequence at every time step.

In [40]:
```
Image(filename='bd.png')
```

Out[40]:

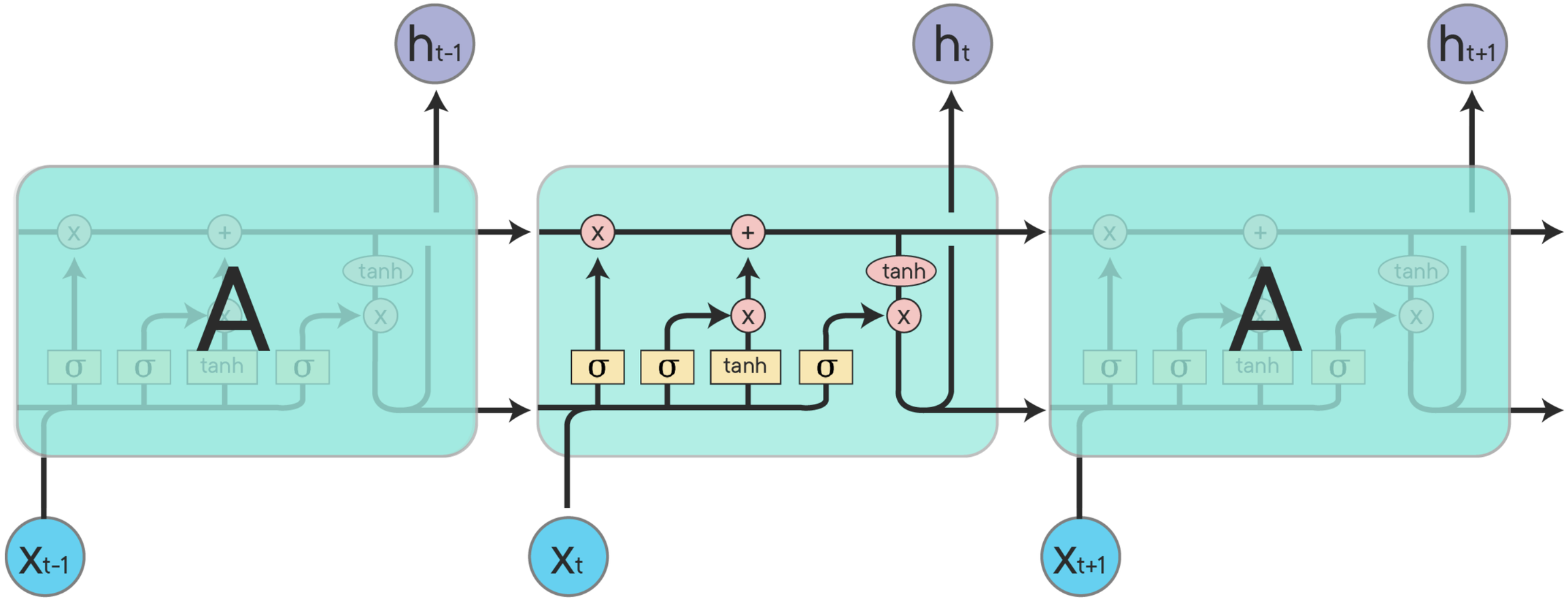


LSTMs are memory cells that use three gates:

- Forget Gate, which determines how much of the current state should be forgotten
- Input Gate, which determines how much should be kept of the cell that was passed
- Output Gate, which determines how much of the current state should be exposed to the next layer in the network

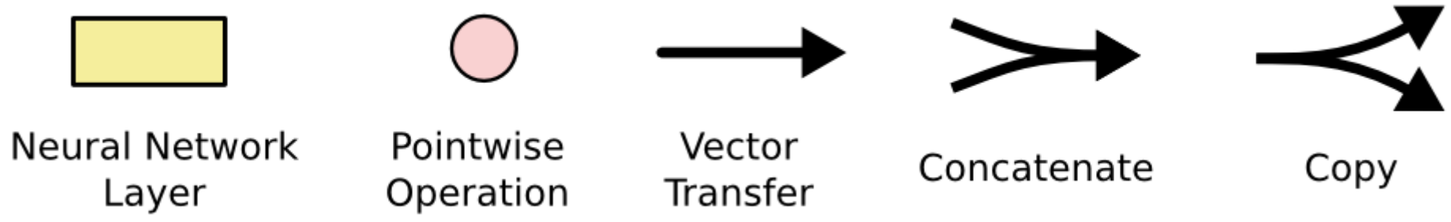
In [34]:
```
Image(filename='LSTM.png')
```

Out[34]:



In [24]:
```
Image(filename='LSTM-notation.png')
```

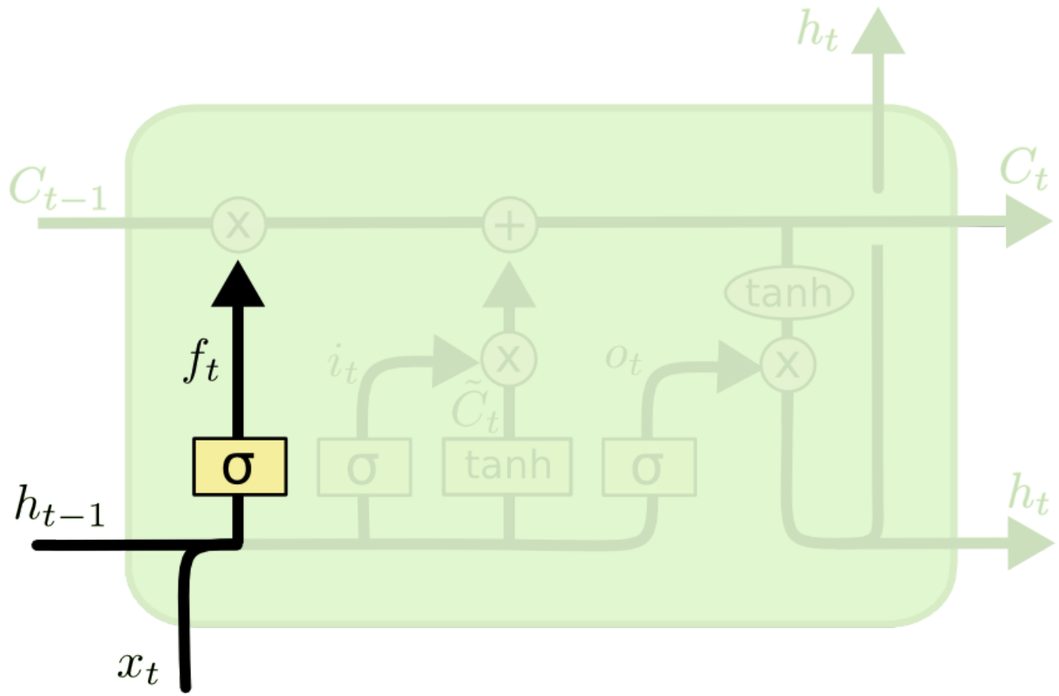
Out[24]:



forget gate activation vector determines how much of the current state should be forgotten with a sigmoid function.

In [27]:
```
Image(filename='LSTM-f.png')
```

Out[27]:

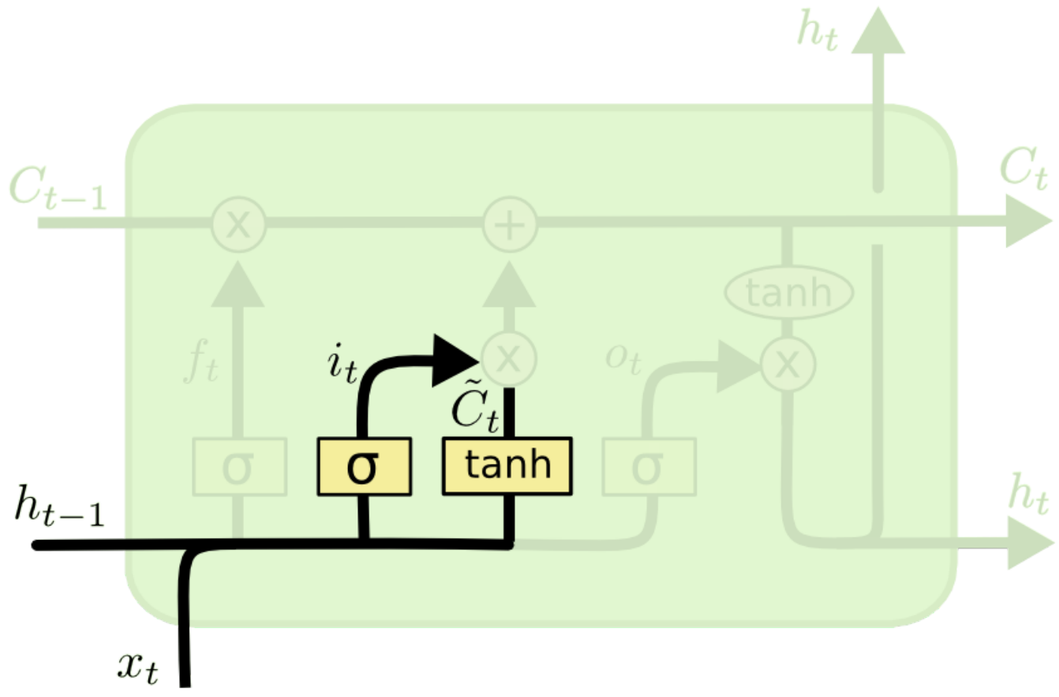


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

New information is stored in the cell state, Ct, in two steps. An input gate layer, sigmoid function, decides which values get updated, and then a tanh layer makes a vector of new candidate values, C̃t, that may be added to the state.

In [28]:
```
Image(filename='LSTM-i.png')
```

Out[28]:



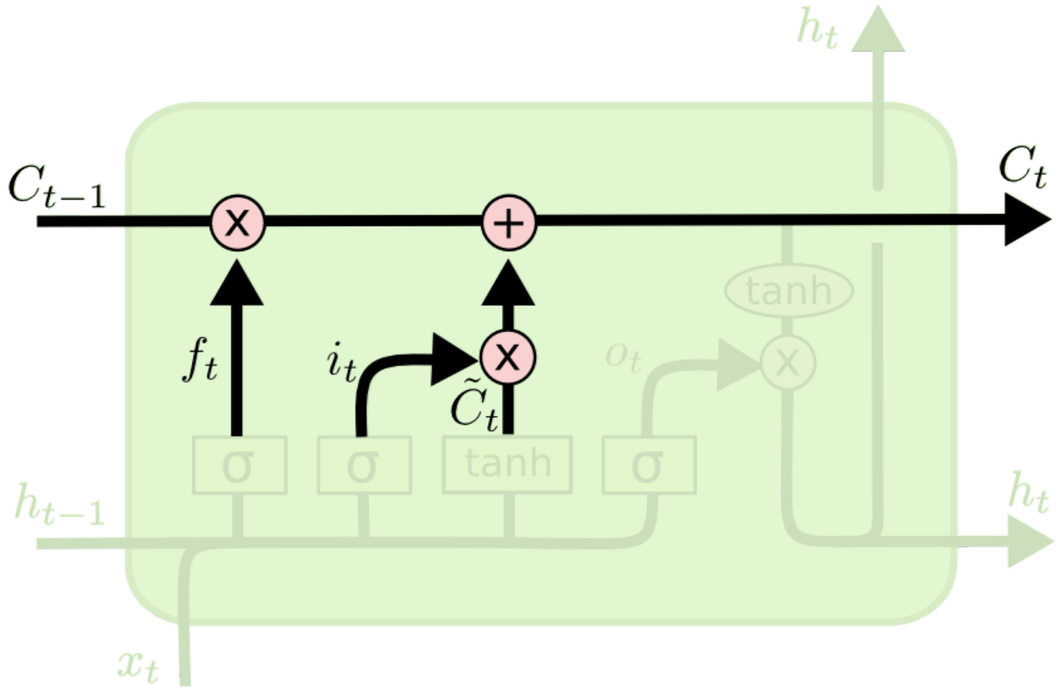
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Update the previous cell state, Ct−1, into the new cell state Ct by multiplying Ct−1 by ft without forgetton elements, and then adding it multiplied by C̃t.

In [29]:
```
Image(filename='LSTM-C.png')
```

Out[29]:

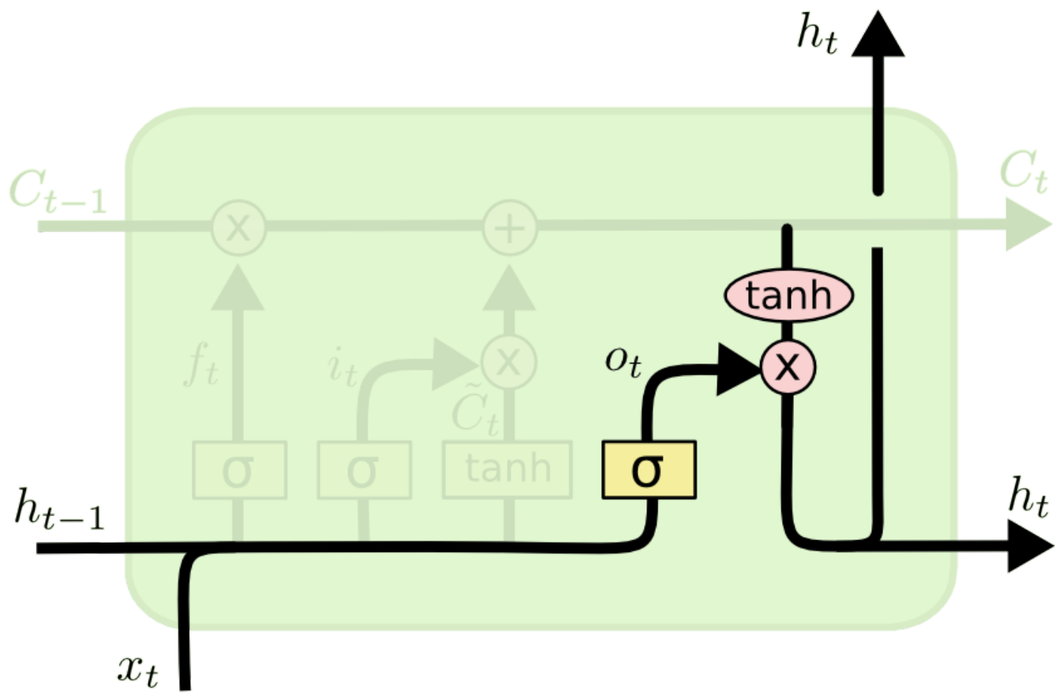


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Determine the output by making a sigmoid layer that decides wich parts of the Ct will be outputed, passing the Ct through tanh to make the values be between −1 and 1, and multiplying the result by the output gate result.

In [30]:
```
Image(filename='LSTM-o.png')
```

Out[30]:



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

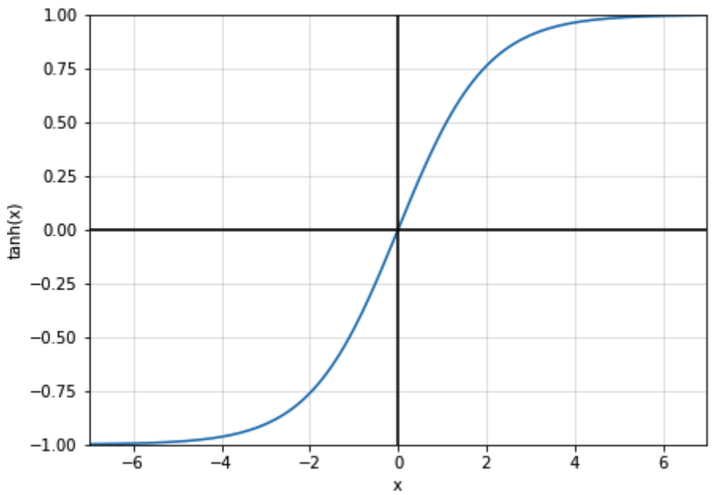
$$h_t = o_t * \tanh\left(C_t\right)$$

Tanh is the hyperbolic tangent function, and is used as an activation function to determine which neurons get passed to the next network layer.

$$tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

In [21]:
```
Image(filename='tanh.png')
```

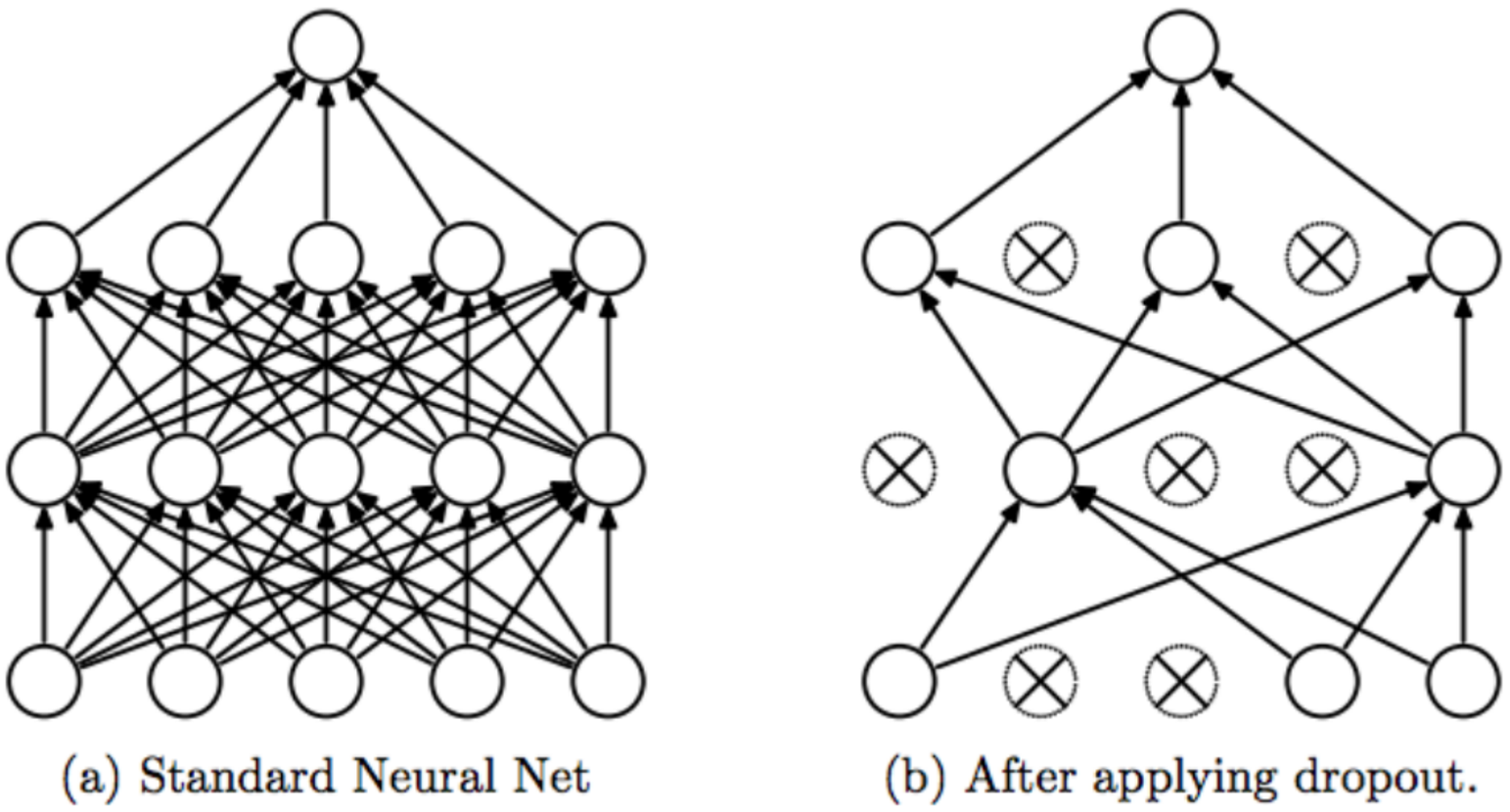
Out[21]:



Dropout is a regularization technique that randomly sets neurons to 0 at each epoch of training.

In [35]:
```
Image(filename='dropout.png')
```

Out[35]:

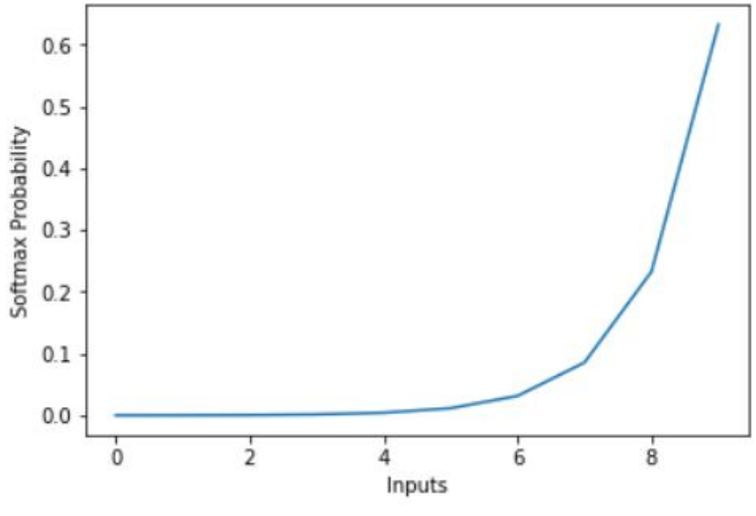


Softmax is an activation function that makes a vector of numbers into a vector of probabilities for each of the 4 classes.

$$softmax = \frac{e^{x_i}}{\sum_j^k e^{x_j}}$$

In [23]:
```
Image(filename='softmax.JPG')
```

Out[23]:



Adaptive Moment Estimation, Adam, computes adaptive learning rates for each parameter by storing an exponentially decaying average of past squared gradients. Adam computes bias-corrected first, m(mean), and second, v(uncentered variance), moment estimates with decay rates, β, that are then used to update the parameters, θ, with an η learning rate.

$$\hat{m} = \frac{m_t}{1 - \beta_{t1}}$$

$$\hat{v} = \frac{v_t}{1 - \beta_{t2}}$$

$$\theta_j^{new} = \theta_j^{previous} - \frac{\eta * \hat{m}_j}{\sqrt{\hat{v}_j} + \varepsilon}$$

Categorical crossentropy is the loss function for softmax, and is the negative log of the softmax function. Optimize the function using the partial derivative of the loss function with repect to x, and then iterate toward optimal loss with an η stepsize.

$$CE = -log(\frac{e^{x_i}}{\sum_j^k e^{x_j}})$$

$$\nabla \frac{\partial CE(x)}{\partial x} = \frac{e^{x_i}}{\sum_j^k e^{x_j}} - 1$$

$$x_j^{new} = x_j^{previous} + \eta * \nabla \frac{\partial CE(x^{previous})}{\partial x^{previous}}$$

The metrics that will be used to determine model performance are accuracy, precision, and recall.

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

In [13]:
```
rnn = Sequential()
rnn.add(Bidirectional(LSTM(units=50, activation='tanh', return_sequences=True, input_shape=(3,4))))
rnn.add(Dropout(0.2))
rnn.add(Dense(units=4, activation='softmax'))
rnn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy', keras.metrics.Precision(), keras.metrics.Recall()])
```

In [14]:
```
history=rnn.fit(X_train, y_train, epochs = 10, batch_size = 20, verbose=1, validation_data=(X_val, y_val))
```

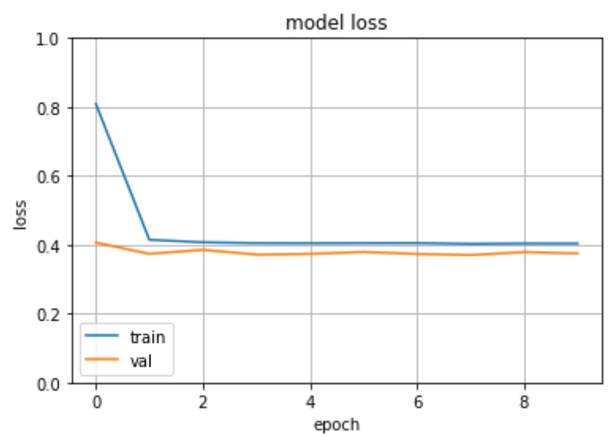
```
Epoch 1/10
350/350 [==============================] - 1s 4ms/step - loss: 0.8087 - accuracy: 0.6668 - precision: 0.8665 - recall: 0.4576 - val_loss: 0.4063 - val_accuracy: 0.8683 - val_precision:
0.9512 - val_recall: 0.7467
Epoch 2/10
350/350 [==============================] - 1s 2ms/step - loss: 0.4143 - accuracy: 0.8180 - precision: 0.9419 - recall: 0.7259 - val_loss: 0.3737 - val_accuracy: 0.8750 - val_precision:
0.9525 - val_recall: 0.7683
Epoch 3/10
350/350 [==============================] - 1s 2ms/step - loss: 0.4071 - accuracy: 0.8173 - precision: 0.9374 - recall: 0.7332 - val_loss: 0.3853 - val_accuracy: 0.8700 - val_precision:
0.9552 - val_recall: 0.7467
Epoch 4/10
350/350 [==============================] - 1s 2ms/step - loss: 0.4047 - accuracy: 0.8186 - precision: 0.9410 - recall: 0.7297 - val_loss: 0.3714 - val_accuracy: 0.8733 - val_precision:
0.9521 - val_recall: 0.7617
Epoch 5/10
350/350 [==============================] - 1s 2ms/step - loss: 0.4044 - accuracy: 0.8180 - precision: 0.9411 - recall: 0.7316 - val_loss: 0.3735 - val_accuracy: 0.8733 - val_precision:
0.9472 - val_recall: 0.7767
Epoch 6/10
350/350 [==============================] - 1s 2ms/step - loss: 0.4048 - accuracy: 0.8170 - precision: 0.9391 - recall: 0.7320 - val_loss: 0.3790 - val_accuracy: 0.8717 - val_precision:
0.9520 - val_recall: 0.7600
Epoch 7/10
350/350 [==============================] - 1s 2ms/step - loss: 0.4050 - accuracy: 0.8187 - precision: 0.9424 - recall: 0.7309 - val_loss: 0.3733 - val_accuracy: 0.8733 - val_precision:
0.9474 - val_recall: 0.7800
Epoch 8/10
350/350 [==============================] - 1s 2ms/step - loss: 0.4026 - accuracy: 0.8195 - precision: 0.9415 - recall: 0.7314 - val_loss: 0.3701 - val_accuracy: 0.8733 - val_precision:
0.9467 - val_recall: 0.7700
Epoch 9/10
350/350 [==============================] - 1s 2ms/step - loss: 0.4037 - accuracy: 0.8188 - precision: 0.9373 - recall: 0.7330 - val_loss: 0.3788 - val_accuracy: 0.8733 - val_precision:
0.9559 - val_recall: 0.7583
Epoch 10/10
350/350 [==============================] - 1s 2ms/step - loss: 0.4035 - accuracy: 0.8193 - precision: 0.9409 - recall: 0.7310 - val_loss: 0.3747 - val_accuracy: 0.8733 - val_precision:
0.9521 - val_recall: 0.7617
```

In [15]:
```
rnn.summary()
```

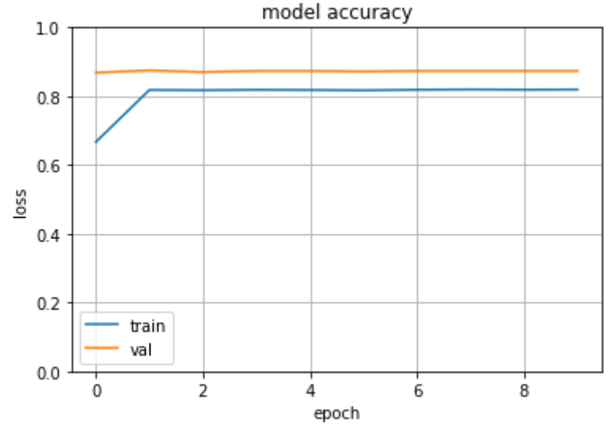
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional (Bidirectional (20, 3, 100)              22000
_________________________________________________________________
dropout (Dropout)            (20, 3, 100)              0
```

```
_________________________________________________________________
dense (Dense)                (20, 3, 4)                404       
=================================================================
Total params: 22,404
Trainable params: 22,404
Non-trainable params: 0
_________________________________________________________________
```

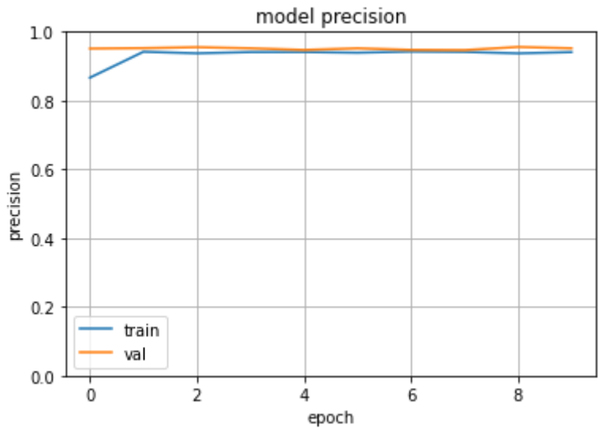
In [16]:
```
rnn_metrics(history.history['loss'],history.history['val_loss'],'model loss','loss')
```



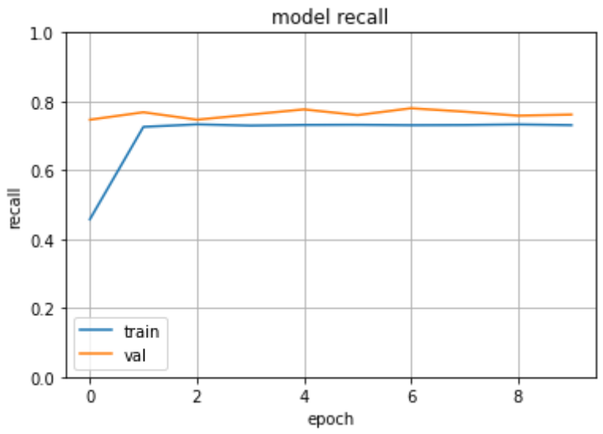
In [17]:
```
rnn_metrics(history.history['accuracy'],history.history['val_accuracy'],'model accuracy','loss')
```



In [18]:
```
rnn_metrics(history.history['precision'],history.history['val_precision'],'model precision','precision')
```



In [19]:
```
rnn_metrics(history.history['recall'],history.history['val_recall'],'model recall','recall')
```



In [25]:
```
predictions=rnn.predict(X_train, verbose=1)
```

```
219/219 [==============================] - 0s 879us/step
```

In [36]:
```
scores = rnn.evaluate(X_test, y_test, batch_size=20, verbose=1)
print(f'Model Accuracy: {round(scores[1]*100,1)}%')
```

```
50/50 [==============================] - 0s 1ms/step - loss: 0.3800 - accuracy: 0.8530 - precision: 0.9325 - recall: 0.7557
Model Accuracy: 85.3%
```

Conclusion

The model is able to predict the next nucleotide of the genome with 85.3% accuracy.

In [ ]:
```

```