1. Write a Python Program to perform brute force attack on the cipher text “dvvkzecfssprkkve"

```python
def decrypt_caesar_cipher(ciphertext, shift):
    """Decrypt the ciphertext using a Caesar cipher with the given shift."""
    plaintext = ""
    for char in ciphertext:
        if char.isalpha():
            # Determine the offset based on character case
            offset = ord('a') if char.islower() else ord('A')
            # Perform the decryption
            decrypted_char = chr((ord(char) - offset - shift) % 26 + offset)
            plaintext += decrypted_char
        else:
            plaintext += char
    return plaintext

def brute_force_caesar_cipher(ciphertext):
    """Try all possible shifts to decrypt the Caesar ciphered text."""
    for shift in range(26):  # Try all possible shifts from 0 to 25
        decrypted_text = decrypt_caesar_cipher(ciphertext, shift)
        print(f"Shift {shift}: {decrypted_text}")

# Ciphertext to be decrypted
ciphertext = "dvvkzecfssprkkve"

# Perform brute-force attack
brute_force_caesar_cipher(ciphertext)
~
```

```
┌──(kali㉿kali)-[~/labPractice]
└─$ python lab5.py
Shift 0: dvvkzecfssprkkve
Shift 1: cuujydberroqjjud
Shift 2: bttixcadqqnpiitc
Shift 3: asshwbzcppmohhsb
Shift 4: zrrgvayboolnggra
Shift 5: yqqfuzxannkmffqz
Shift 6: xppetywzmmjleepy
Shift 7: woodsxvyllikddox
Shift 8: vnncrwuxkkhjccnw
Shift 9: ummbqvtwjjgibbmv
Shift 10: tllapusviifhaalu
Shift 11: skkzotruhhegzzkt
Shift 12: rjjynsqtggdfyyjs
Shift 13: qiixmrpsffcexxir
Shift 14: phhwlqoreebdwwhq
Shift 15: oggvkpnqddacvvgp
Shift 16: nffujompcczbuufo
Shift 17: meetinlobbyatten
Shift 18: lddshmknaaxzssdm
Shift 19: kccrgljmzzwyrrcl
Shift 20: jbbqfkilyyvxqqbk
Shift 21: iaapejhkxxuwppaj
Shift 22: hzzodigjwwtvoozi
Shift 23: gyynchfivvsunnyh
Shift 24: fxxmbgehuurtmmxg
Shift 25: ewwlafdgttqsllwf
```

2. Write a Python program to use brute force attack to decipher the message.

Assume Affine cipher was used and "ab" is encrypted as "GL". Find the value of keys.

XPALASXYFGFUKPXUSOGEUTKCDGFXANMGNVS

```python
#!/usr/bin/python
def mod_inverse(a, m):
    """ Return the modular inverse of a under modulo m """
    a = a % m
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    raise ValueError("Modular inverse does not exist")

def affine_decrypt(ciphertext, a, b):
    """ Decrypt ciphertext using Affine cipher with keys a and b """
    m = 26
    a_inv = mod_inverse(a, m)
    plaintext = ""
    for char in ciphertext:
        if char.isalpha():
            # Convert character to 0-25 range
            y = ord(char) - ord('A')
            # Decrypt
            x = (a_inv * (y - b)) % m
            # Convert back to letter
            plaintext += chr(x + ord('A'))
        else:
            plaintext += char
    return plaintext

def brute_force_affine(ciphertext):
    """ Try all possible values for a and b to decrypt the message """
    m = 26
    for a in range(1, m):
        if gcd(a, m) == 1:  # a must be coprime with 26
            for b in range(m):
                decrypted_text = affine_decrypt(ciphertext, a, b)
                print(f"a={a}, b={b}: {decrypted_text}")
```

```python
def gcd(a, b):
    """ Compute the greatest common divisor of a and b """
    while b:
        a, b = b, a % b
    return a

# Ciphertext to be decrypted
ciphertext = "XPALASXYFGFUKPXUSOGEUTKCDGFXANMGNVS"

# Perform brute-force attack
brute_force_affine(ciphertext)
```

```
┌──(kali㉿kali)-[~/labPractice]
└─$ python lab5.py
a=1, b=0: XPALASXYFGFUKPXUSOGEUTKCDGFXANMGNVS
a=1, b=1: WOZKZRWXEFETJOWTRNFDTSJBCFEWZMLFMUR
a=1, b=2: VNYJYQVWDEDSINVSQMECSRIABEDVYLKELTQ
a=1, b=3: UMXIXPUVCDCRHMURPLDBRQHZADCUXKJDKSP
a=1, b=4: TLWHWOTUBCBQGLTQOKCAQPGYZCBTWJICJRO
a=1, b=5: SKVGVNSTABAPFKSPNJBZPOFXYBASVIHBIQN
a=1, b=6: RJUFUMRSZAZOEJROMIAYONEWXAZRUHGAHPM
a=1, b=7: QITETLQRYZYNDIQNLHZXNMDVWZYQTGFZGOL
a=1, b=8: PHSDSKPQXYXMCHPMKGYWMLCUVYXPSFEYFNK
a=1, b=9: OGRCRJOPWXWLBGOLJFXVLKBTUXWOREDXEMJ
a=1, b=10: NFQBQINOVWVKAFNKIEWUKJASTWVNQDCWDLI
a=1, b=11: MEPAPHMNUVUJZEMJHDVTJIZRSVUMPCBVCKH
a=1, b=12: LDOZOGLMTUTIYDLIGCUSIHYQRUTLOBAUBJG
a=1, b=13: KCNYNFKLSTSHXCKHFBTRHGXPQTSKNAZTAIF
a=1, b=14: JBMXMEJKRSRGWBJGEASQGFWOPSRJMZYSZHE
a=1, b=15: IALWLDIJQRQFVAIFDZRPFEVNORQILYXRYGD
a=1, b=16: HZKVKCHIPQPEUZHECYQOEDUMNQPHKXWQXFC
a=1, b=17: GYJUJBGHOPODTYGDBXPNDCTLMPOGJWVPWEB
a=1, b=18: FXITIAFGNONCSXFCAWOMCBSKLONFIVUOVDA
a=1, b=19: EWHSHZEFMNMBRWEBZVNLBARJKNMEHUTNUCZ
a=1, b=20: DVGRGYDELMLAQVDAYUMKAZQIJMLDGTSMTBY
a=1, b=21: CUFQFXCDKLKZPUCZXTLJZYPHILKCFSRLSAX
a=1, b=22: BTEPEWBCJKJYOTBYWSKIYXOGHKJBERQKRZW
a=1, b=23: ASDODVABIJIXNSAXVRJHXWNFGJIADQPJQYV
a=1, b=24: ZRCNCUZAHIHWMRZWUQIGWVMEFIHZCPOIPXU
a=1, b=25: YQBMBTYZGHGVLQYVTPHFVULDEHGYBONHOWT
a=3, b=0: ZFAVAGZITCTYMFZYGWCKYPMSBCTZANECNHG
a=3, b=1: QWRMRXQZKTKPDWQPXNTBPGDJSTKQREVTEYX
a=3, b=2: HNIDIOHQBKBGUNHGOEKSGXUAJKBHIVMKVPO
a=3, b=3: YEZUZFYHSBSXLEYXFVBJXOLRABSYZMDBMGF
a=3, b=4: PVQLQWPYJSJOCVPOWMSAOFCIRSJPQDUSDXW
a=3, b=5: GMHCHNGPAJAFTMGFNDJRFWTZIJAGHULJUON
a=3, b=6: XDYTYEXGRARWKDXWEUAIWNKQZARXYLCALFE
a=3, b=7: OUPKPVOXIRINBUONVLRZNEBHQRIOPCTRCWV
a=3, b=8: FLGBGMFOZIZESLFEMCIQEVSYHIZFGTKITNM
a=3, b=9: WCXSXDWFQZQVJCWVDTZHVMJPYZQWXKBZKED
```

```
a=3, b=10: NTOJOUNWHQHMATNMUKQYMDAGPQHNOBSQBVU
a=3, b=11: EKFAFLENYHYDRKEDLBHPDURXGHYEFSJHSML
a=3, b=12: VBWRWCVEPYPUIBVUCSYGULIOXYPVWJAYJDC
a=3, b=13: MSNINTMVGPGLZSMLTJPXLCZFOPGMNARPAUT
a=3, b=14: DJEZEKDMXGXCQJDCKAGOCTQWFGXDERIGRLK
a=3, b=15: UAVQVBUDOXOTHAUTBRXFTKHNWXOUVIZXICB
a=3, b=16: LRMHMSLUFOFKYRLKSIOWKBYENOFLMZQOZTS
a=3, b=17: CIDYDJCLWFWBPICBJZFNBSPVEFWCDQHFQKJ
a=3, b=18: TZUPUATCNWNSGZTSAQWESJGMVWNTUHYWHBA
a=3, b=19: KQLGLRKTENEJXQKJRHNVJAXDMNEKLYPNYSR
a=3, b=20: BHCXCIBKVEVAOHBAIYEMAROUDEVBCPGEPJI
a=3, b=21: SYTOTZSBMVMRFYSRZPVDRIFLUVMSTGXVGAZ
a=3, b=22: JPKFKQJSDMDIWPJIQGMUIZWCLMDJKXOMXRQ
a=3, b=23: AGBWBHAJUDUZNGAZHXDLZQNTCDUABOFDOIH
a=3, b=24: RXSNSYRALULQEXRQYOUCQHEKTULRSFWUFZY
a=3, b=25: IOJEJPIRCLCHVOIHPFLTHYVBKLCIJWNLWQP
a=5, b=0: PDAXAOPKBWBECDPEOIWGEJCQLWBPANSWNZO
a=5, b=1: UIFCFTUPGBGJHIUJTNBLJOHVQBGUFSXBSET
a=5, b=2: ZNKHKYZULGLOMNZOYSGQOTMAVGLZKXCGXJY
a=5, b=3: ESPMPDEZQLQTRSETDXLVTYRFALQEPCHLCOD
a=5, b=4: JXURUIJEVQVYWXJYICQAYDWKFQVJUHMQHTI
a=5, b=5: OCZWZNOJAVADBCODNHVFDIBPKVAOZMRVMYN
a=5, b=6: THEBESTOFAFIGHTISMAKINGUPAFTERWARDS
a=5, b=7: YMJGJXYTKFKNLMYNXRFPNSLZUFKYJWBFWIX
a=5, b=8: DROLOCDYPKPSQRDSCWKUSXQEZKPDOBGKBNC
a=5, b=9: IWTQTHIDUPUXVWIXHBPZXCVJEPUITGLPGSH
a=5, b=10: NBYVYMNIZUZCABNCMGUECHAOJUZNYLQULXM
a=5, b=11: SGDADRSNEZEHFGSHRLZJHMFTOZESDQVZQCR
a=5, b=12: XLIFIWXSJEJMKLXMWQEOMRKYTEJXIVAEVHW
a=5, b=13: CQNKNBCXOJORPQCRBVJTRWPDYJOCNAFJAMB
a=5, b=14: HVSPSGHCTOTWUVHWGAOYWBUIDOTHSFKOFRG
a=5, b=15: MAXUXLMHYTYBZAMBLFTDBGZNITYMXKPTKWL
a=5, b=16: RFCZCQRMDYDGEFRGQKYIGLESNYDRCPUYPBQ
a=5, b=17: WKHEHVWRIDILJKWLVPDNLQJXSDIWHUZDUGV
a=5, b=18: BPMJMABWNINQOPBQAUISQVOCXINBMZEIZLA
a=5, b=19: GURORFGBSNSVTUGVFZNXVATHCNSGREJNEQF
```

```
a=5, b=19: GURORFGBSNSVTUGVFZNXVATHCNSGREJNEQF
a=5, b=20: LZWTWKLGXSXAYZLAKESCAFYMHSXLWJOSJVK
a=5, b=21: QEBYBPQLCXCFDEQFPJXHFKDRMXCQBOTXOAP
a=5, b=22: VJGDGUVQHCHKIJVKUOCMKPIWRCHVGTYCTFU
a=5, b=23: AOLILZAVMHMPNOAPZTHRPUNBWHMALYDHYKZ
a=5, b=24: FTQNQEFARMRUSTFUEYMWUZSGBMRFQDIMDPE
a=5, b=25: KYVSVJKFWRWZXYKZJDRBZEXLGRWKVINRIUJ
a=7, b=0: HRAJAKHWXMXOURHOKCMIOZUETMXHANYMNDK
a=7, b=1: SCLULVSHIXIZFCSZVNXTZKFPEXISLYJXYOV
a=7, b=2: DNWFWGDSTITKQNDKGYIEKVQAPITDWJUIJZG
a=7, b=3: OYHQHRODETEVBYOVRJTPVGBLATEOHUFTUKR
a=7, b=4: ZJSBSCZOPEPGMJZGCUEAGRMWLEPZSFQEFVC
a=7, b=5: KUDMDNKZAPARXUKRNFPLRCXHWPAKDQBPQGN
a=7, b=6: VFOXOYVKLALCIFVCYQAWCNISHALVOBMABRY
a=7, b=7: GQZIZJGVWLWNTQGNJBLHNYTDSLWGZMXLMCJ
a=7, b=8: RBKTKURGHWHYEBRYUMWSYJEODWHRKXIWXNU
a=7, b=9: CMVEVFCRSHSJPMCJFXHDJUPZOHSCVITHIYF
a=7, b=10: NXGPGQNCDSDUAXNUQISOUFAKZSDNGTESTJQ
a=7, b=11: YIRARBYNODOFLIYFBTDZFQLVKDOYREPDEUB
a=7, b=12: JTCLCMJYZOZQWTJQMEOKQBWGVOZJCPAOPFM
a=7, b=13: UENWNXUJKZKBHEUBXPZVBMHRGZKUNALZAQX
a=7, b=14: FPYHYIFUVKVMSPFMIAKGMXSCRKVFYLWKLBI
a=7, b=15: QAJSJTQFGVGXDAQXTLVRXIDNCVGQJWHVWMT
a=7, b=16: BLUDUEBQRGRIOLBIEWGCITOYNGRBUHSGHXE
a=7, b=17: MWFOFPMBCRCTZWMTPHRNTEZJYRCMFSDRSIP
a=7, b=18: XHQZQAXMNCNEKHXEASCYEPKUJCNXQDOCDTA
a=7, b=19: ISBKBLIXYNYPVSIPLDNJPAVFUNYIBOZNOEL
a=7, b=20: TDMVMWTIJYJAGDTAWOYUALGQFYJTMZKYZPW
a=7, b=21: EOXGXHETUJULROELHZJFLWRBQJUEXKVJKAH
a=7, b=22: PZIRISPEFUFWCZPWSKUQWHCMBUFPIVGUVLS
a=7, b=23: AKTCTDAPQFQHNKAHDVFBHSNXMFQATGRFGWD
a=7, b=24: LVENEOLABQBSYVLSOGQMSDYIXQBLERCQRHO
a=7, b=25: WGPYPZWLMBMDJGWDZRBXDOJTIBMWPCNBCSZ
a=9, b=0: RTAHACRUPSPIETRICQSMIFEGJSPRANKSNLC
a=9, b=1: OQXEXZORMPMFBQOFZNPJFCBDGPMOXKHPKIZ
a=9, b=2: LNUBUWLOJMJCYNLCWKMGCZYADMJLUHEMHFW
a=9, b=3: IKRYRTILGJGZVKIZTHJDZWVXAJGIREBJECT
a=9, b=4: FHOVOQFIDGDWSHFWQEGAWTSUXGDFOBYGBZQ
a=9, b=5: CELSLNCFADATPECTNBDXTQPRUDACLYVDYWN
a=9, b=6: ZBIPIKZCXAXQMBZQKYAUQNMORAXZIVSAVTK
```

```
a=9, b=6: ZBIPIKZCXAXQMBZQKYAUQNMORAXZIVSAVTK
a=9, b=7: WYFMFHWZUXUNJYWNHVXRNKJLOXUWFSPXSQH
a=9, b=8: TVCJCETWRURKGVTKESUOKHGILURTCPMUPNE
a=9, b=9: QSZGZBQTOROHDSQHBPRLHEDFIROQZMJRMKB
a=9, b=10: NPWDWYNQLOLEAPNEYMOIEBACFOLNWJGOJHY
a=9, b=11: KMTATVKNILIBXMKBVJLFBYXZCLIKTGDLGEV
a=9, b=12: HJQXQSHKFIFYUJHYSGICYVUWZIFHQDAIDBS
a=9, b=13: EGNUNPEHCFCVRGEVPDFZVSRTWFCENAXFAYP
a=9, b=14: BDKRKMBEZCZSODBSMACWSPOQTCZBKXUCXVM
a=9, b=15: YAHOHJYBWZWPLAYPJXZTPMLNQZWYHURZUSJ
a=9, b=16: VXELEGVYTWTMIXVMGUWQMJIKNWTVEROWRPG
a=9, b=17: SUBIBDSVQTQJFUSJDRTNJGFHKTQSBOLTOMD
a=9, b=18: PRYFYAPSNQNGCRPGAOQKGDCEHQNPYLIQLJA
a=9, b=19: MOVCVXMPKNKDZOMDXLNHDAZBENKMVIFNIGX
a=9, b=20: JLSZSUJMHKHAWLJAUIKEAXWYBKHJSFCKFDU
a=9, b=21: GIPWPRGJEHEXTIGXRFHBXUTVYHEGPCZHCAR
a=9, b=22: DFMTMODGBEBUQFDUOCEYURQSVEBDMZWEZXO
a=9, b=23: ACJQJLADYBYRNCARLZBVRONPSBYAJWTBWUL
a=9, b=24: XZGNGIXAVYVOKZXOIWYSOLKMPYVXGTQYTRI
a=9, b=25: UWDKDFUXSVSLHWULFTVPLIHJMVSUDQNVQOF
a=11, b=0: VZABAEVORKRQIZVQEGKYQXIMFKRVANUKNJE
a=11, b=1: CGHIHLCVYRYXPGCXLNRFXEPTMRYCHUBRUQL
a=11, b=2: JNOPOSJCFYFEWNJESUYMELWATYFJOBIYBXS
a=11, b=3: QUVWVZQJMFMLDUQLZBFTLSDHAFMQVIPFIEZ
a=11, b=4: XBCDCGXQTMTSKBXSGIMASZKOHMTXCPWMPLG
a=11, b=5: EIJKJNEXATAZRIEZNPTHZGRVOTAEJWDTWSN
a=11, b=6: LPQRQULEHAHGYPLGUWAOGNYCVAHLQDKADZU
a=11, b=7: SWXYXBSLOHONFWSNBDHVNUFJCHOSXKRHKGB
a=11, b=8: ZDEFEIZSVOVUMDZUIKOCUBMQJOVZERYORNI
a=11, b=9: GKLMLPGZCVCBTKGBPRVJBITXQVCGLYFVYUP
a=11, b=10: NRSTSWNGJCJIARNIWYCQIPAEXCJNSFMCFBW
a=11, b=11: UYZAZDUNQJQPHYUPDFJXPWHLEJQUZMTJMID
a=11, b=12: BFGHGKBUXQXWOFBWKMQEWDOSLQXBGTAQTPK
a=11, b=13: IMNONRIBEXEDVMIDRTXLDKVZSXEINAHXAWR
a=11, b=14: PTUVUYPILELKCTPKYAESKRCGZELPUHOEHDY
a=11, b=15: WABCBFWPSLSRJAWRFHLZRYJNGLSWBOVLOKF
a=11, b=16: DHIJIMDWZSZYQHDYMOSGYFQUNSZDIVCSVRM
a=11, b=17: KOPQPTKDGZGFXOKFTVZNFMXBUZGKPCJZCYT
a=11, b=18: RVWXWARKNGNMEVRMACGUMTEIBGNRWJQGJFA
a=11, b=19: YCDEDHYRUNUTLCYTHJNBTALPINUYDQXNQMH
```

```
a=11, b=19: YCDEDHYRUNUTLCYTHJNBTALPINUYDQXNQMH
a=11, b=20: FJKLKOFYBUBASJFAOQUIAHSWPUBFKXEUXTO
a=11, b=21: MQRSRVMFIBIHZQMHVXBPHOZDWBIMRELBEAV
a=11, b=22: TXYZYCTMPIPOGXTOCEIWOVGKDIPTYLSILHC
a=11, b=23: AEFGFJATWPWVNEAVJLPDVCNRKPWAFSZPSOJ
a=11, b=24: HLMNMQHADWDCULHCQSWKCJUYRWDHMZGWZVQ
a=11, b=25: OSTUTXOHKDKJBSOJXZDRJQBFYDKOTGNDGCX
a=15, b=0: FBAZAWFMJQJKSBFKWUQCKDSOVQJFANGQNRW
a=15, b=1: YUTSTPYFCJCDLUYDPNJVDWLHOJCYTGZJGKP
a=15, b=2: RNMLMIRYVCVWENRWIGCOWPEAHCVRMZSCZDI
a=15, b=3: KGFEFBKROVOPXGKPBZVHPIXTAVOKFSLVSWB
a=15, b=4: DZYXYUDKHOHIQZDIUSOAIBQMTOHDYLEOLPU
a=15, b=5: WSRQRNWDAHABJSWBNLHTBUJFMHAWREXHEIN
a=15, b=6: PLKJKGPWTATUCLPUGEAMUNCYFATPKXQAXBG
a=15, b=7: IEDCDZIPMTMNVEINZXTFNGVRYTMIDQJTQUZ
a=15, b=8: BXWVWSBIFMFGOXBGSQMYGZOKRMFBWJCMJNS
a=15, b=9: UQPOPLUBYFYZHQUZLJFRZSHDKFYUPCVFCGL
a=15, b=10: NJIHIENURYRSAJNSECYKSLAWDYRNIVOYVZE
a=15, b=11: GCBABXGNKRKLTCGLXVRDLETPWRKGBOHROSX
a=15, b=12: ZVUTUQZGDKDEMVZEQOKWEXMIPKDZUHAKHLQ
a=15, b=13: SONMNJSZWDWXFOSXJHDPXQFBIDWSNATDAEJ
a=15, b=14: LHGFGCLSPWPQYHLQCAWIQJYUBWPLGTMWTXC
a=15, b=15: EAZYZVELIPIJRAEJVTPBJCRNUPIEZMFPMQV
a=15, b=16: XTSRSOXEBIBCKTXCOMIUCVKGNIBXSFYIFJO
a=15, b=17: QMLKLHQXUBUVDMQVHFBNVODZGBUQLYRBYCH
a=15, b=18: JFEDEAJQNUNOWFJOAYUGOHWSZUNJERKURVA
a=15, b=19: CYXWXTCJGNGHPYCHTRNZHAPLSNGCXKDNKOT
a=15, b=20: VRQPQMVCZGZAIRVAMKGSATIELGZVQDWGDHM
a=15, b=21: OKJIJFOVSZSTBKOTFDZLTMBXEZSOJWPZWAF
a=15, b=22: HDCBCYHOLSLMUDHMYWSEMFUQXSLHCPISPTY
a=15, b=23: AWVUVRAHELEFNWAFRPLXFYNJQLEAVIBLIMR
a=15, b=24: TPONOKTAXEXYGPTYKIEQYRGCJEXTOBUEBFK
a=15, b=25: MIHGHDMTQXQRZIMRDBXJRKZVCXQMHUNXUYD
a=17, b=0: JHATAYJGLILSWHJSYKIOSVWURILJANQINPY
a=17, b=1: MKDWDBMJOLOVZKMVBNLRVYZXULOMDQTLQSB
a=17, b=2: PNGZGEPMRORYCNPYEQOUYBCAXORPGTWOTVE
a=17, b=3: SQJCJHSPURUBFQSBHTRXBEFDARUSJWZRWYH
a=17, b=4: VTMFMKVSXUXEITVEKWUAEHIGDUXVMZCUZBK
a=17, b=5: YWPIPNYVAXAHLWYHNZXDHKLJGXAYPCFXCEN
a=17, b=6: BZSLSQBYDADKOZBKQCAGKNOMJADBSFIAFHQ
```

```
a=17, b=6: BZSLSQBYDADKOZBKQCAGKNOMJADBSFIAFHQ
a=17, b=7: ECVOVTEBGDGNRCENTFDJNQRPMDGEVILDIKT
a=17, b=8: HFYRYWHEJGJQUFHQWIGMQTUSPGJHYLOGLNW
a=17, b=9: KIBUBZKHMJMTXIKTZLJPTWXVSJMKBORJOQZ
a=17, b=10: NLEXECNKPMPWALNWCOMSWZAYVMPNERUMRTC
a=17, b=11: QOHAHFQNSPSZDOQZFRPVZCDBYPSQHUXPUWF
a=17, b=12: TRKDKITQVSVCGRTCIUSYCFGEBSVTKXASXZI
a=17, b=13: WUNGNLWTYVYFJUWFLXVBFIJHEVYWNADVACL
a=17, b=14: ZXQJQOZWBYBIMXZIOAYEILMKHYBZQDGYDFO
a=17, b=15: CATMTRCZEBELPACLRDBHLOPNKBECTGJBGIR
a=17, b=16: FDWPWUFCHEHOSDFOUGEKORSQNEHFWJMEJLU
a=17, b=17: IGZSZXIFKHKRVGIRXJHNRUVTQHKIZMPHMOX
a=17, b=18: LJCVCALINKNUYJLUAMKQUXYWTKNLCPSKPRA
a=17, b=19: OMFYFDOLQNQXBMOXDPNTXABZWNQOFSVNSUD
a=17, b=20: RPIBIGROTQTAEPRAGSQWADECZQTRIVYQVXG
a=17, b=21: USLELJURWTWDHSUDJVTZDGHFCTWULYBTYAJ
a=17, b=22: XVOHOMXUZWZGKVXGMYWCGJKIFWZXOBEWBDM
a=17, b=23: AYRKRPAXCZCJNYAJPBZFJMNLIZCAREHZEGP
a=17, b=24: DBUNUSDAFCFMQBDMSECIMPQOLCFDUHKCHJS
a=17, b=25: GEXQXVGDIFIPTEGPVHFLPSTROFIGXKNFKMV
a=19, b=0: TJARAQTEDODMGJTMQYOSMBGWHODTANCONXQ
a=19, b=1: IYPGPFITSDSBVYIBFNDHBQVLWDSIPCRDCMF
a=19, b=2: XNEVEUXIHSHQKNXQUCSWQFKALSHXERGSRBU
a=19, b=3: MCTKTJMXWHWFZCMFJRHLFUZPAHWMTGVHGQJ
a=19, b=4: BRIZIYBMLWLUORBUYGWAUJOEPWLBIVKWVFY
a=19, b=5: QGXOXNQBALAJDGQJNVLPJYDTELAQXKZLKUN
a=19, b=6: FVMDMCFQPAPYSVFYCKAEYNSITAPFMZOAZJC
a=19, b=7: UKBSBRUFEPENHKUNRZPTNCHXIPEUBODPOYR
a=19, b=8: JZQHQGJUTETCWZJCGOEICRWMXETJQDSEDNG
a=19, b=9: YOFWFVYJITIRLOYRVDTXRGLBMTIYFSHTSCV
a=19, b=10: NDULUKNYXIXGADNGKSIMGVAQBIXNUHWIHRK
a=19, b=11: CSJAJZCNMXMVPSCVZHXBVKPFQXMCJWLXWGZ
a=19, b=12: RHYPYORCBMBKEHRKOWMQKZEUFMBRYLAMLVO
a=19, b=13: GWNENDGRQBQZTWGZDLBFZOTJUBQGNAPBAKD
a=19, b=14: VLCTCSVGFQFOILVOSAQUODIYJQFVCPEQPZS
a=19, b=15: KARIRHKVUFUDXAKDHPFJDSXNYFUKRETFEOH
a=19, b=16: ZPGXGWZKJUJSMPZSWEUYSHMCNUJZGTIUTDW
a=19, b=17: OEVMVLOZYJYHBEOHLTJNHWBRCJYOVIXJISL
a=19, b=18: DTKBKADONYNWQTDWAIYCWLQGRYNDKXMYXHA
a=19, b=19: SIZQZPSDCNCLFISLPXNRLAFVGNCSZMBNMWP
```

```
a=19, b=19: SIZQZPSDCNCLFISLPXNRLAFVGNCSZMBNMWP
a=19, b=20: HXOFOEHSRCRAUXHAEMCGAPUKVCRHOBQCBLE
a=19, b=21: WMDUDTWHGRGPJMWPTBRVPEJZKRGWDQFRQAT
a=19, b=22: LBSJSILWVGVEYBLEIQGKETYOZGVLSFUGFPI
a=19, b=23: AQHYHXALKVKTNQATXFVZTINDOVKAHUJVUEX
a=19, b=24: PFWNWMPAZKZICFPIMUKOIXCSDKZPWJYKJTM
a=19, b=25: EULCLBEPOZOXRUEXBJZDXMRHSZOELYNZYIB
a=21, b=0: LXADAMLQZEZWYXLWMSEUWRYKPEZLANIENBM
a=21, b=1: GSVYVHGLUZURTSGRHNZPRMTFKZUGVIDZIWH
a=21, b=2: BNQTQCBGPUPMONBMCIUKMHOAFUPBQDYUDRC
a=21, b=3: WILOLXWBKPKHJIWHXDPFHCJVAPKWLYTPYMX
a=21, b=4: RDGJGSRWFKFCEDRCSYKACXEQVKFRGTOKTHS
a=21, b=5: MYBEBNMRAFAXZYMXNTFVXSZLQFAMBOJFOCN
a=21, b=6: HTWZWIHMVAVSUTHSIOAQSNUGLAVHWJEAJXI
a=21, b=7: CORURDCHQVQNPOCNDJVLNIPBGVQCREZVESD
a=21, b=8: XJMPMYXCLQLIKJXIYEQGIDKWBQLXMZUQZNY
a=21, b=9: SEHKHTSXGLGDFESDTZLBDYFRWLGSHUPLUIT
a=21, b=10: NZCFCONSBGBYAZNYOUGWYTAMRGBNCPKGPDO
a=21, b=11: IUXAXJINWBWTVUITJPBRTOVHMBWIXKFBKYJ
a=21, b=12: DPSVSEDIRWROQPDOEKWMOJQCHWRDSFAWFTE
a=21, b=13: YKNQNZYDMRMJLKYJZFRHJELXCRMYNAVRAOZ
a=21, b=14: TFILIUTYHMHEGFTEUAMCEZGSXMHTIVQMVJU
a=21, b=15: OADGDPOTCHCZBAOZPVHXZUBNSHCODQLHQEP
a=21, b=16: JVYBYKJOXCXUWVJUKQCSUPWINCXJYLGCLZK
a=21, b=17: EQTWTFEJSXSPRQEPFLXNPKRDIXSETGBXGUF
a=21, b=18: ZLOROAZENSNKMLZKAGSIKFMYDSNZOBWSBPA
a=21, b=19: UGJMJVUZINIFHGUFVBNDFAHTYNIUJWRNWKV
a=21, b=20: PBEHEQPUDIDACBPAQWIYAVCOTIDPERMIRFQ
a=21, b=21: KWZCZLKPYDYVXWKVLRDTVQXJODYKZMHDMAL
a=21, b=22: FRUXUGFKTYTQSRFQGMYOQLSEJYTFUHCYHVG
a=21, b=23: AMPSPBAFOTOLNMALBHTJLGNZETOAPCXTCQB
a=21, b=24: VHKNKWVAJOJGIHVGWCOEGBIUZOJVKXSOXLW
a=21, b=25: QCFIFRQVEJEBDCQBRXJZBWDPUJEQFSNJSGR
a=23, b=0: BVAFAUBSHYHCOVBCUEYQCLOIZYHBANWYNTU
a=23, b=1: KEJOJDKBQHQLXEKLDNHZLUXRIHQKJWFHWCD
a=23, b=2: TNSXSMTKZQZUGNTUMWQIUDGARQZTSFOQFLM
a=23, b=3: CWBGBVCTIZIDPWCDVFZRDMPJAZICBOXZOUV
a=23, b=4: LFKPKELCRIRMYFLMEOIAMVYSJIRLKXGIXDE
a=23, b=5: UOTYTNULARAVHOUVNXRJVEHBSRAUTGPRGMN
a=23, b=6: DXCHCWDUJAJEQXDEWGASENQKBAJDCPYAPVW
```

```
a=23, b=6: DXCHCWDUJAJEQXDEWGASENQKBAJDCPYAPVW
a=23, b=7: MGLQLFMDSJSNZGMNFPJBNWZTKJSMLYHJYEF
a=23, b=8: VPUZUOVMBSBWIPVWOYSKWFICTSBVUHQSHNO
a=23, b=9: EYDIDXEVKBKFRYEFXHBTFORLCBKEDQZBQWX
a=23, b=10: NHMRMGNETKTOAHNOGQKCOXAULKTNMZIKZFG
a=23, b=11: WQVAVPWNCTCXJQWXPZTLXGJDUTCWVIRTIOP
a=23, b=12: FZEJEYFWLCLGSZFGYICUGPSMDCLFERACRXY
a=23, b=13: OINSNHOFULUPBIOPHRLDPYBVMLUONAJLAGH
a=23, b=14: XRWBWQXODUDYKRXYQAUMYHKEVUDXWJSUJPQ
a=23, b=15: GAFKFZGXMDMHTAGHZJDVHQTNEDMGFSBDSYZ
a=23, b=16: PJOTOIPGVMVQCJPQISMEQZCWNMVPOBKMBHI
a=23, b=17: YSXCXRYPEVEZLSYZRBVNZILFWVEYXKTVKQR
a=23, b=18: HBGLGAHYNENIUBHIAKEWIRUOFENHGTCETZA
a=23, b=19: QKPUPJQHWNWRDKQRJTNFRADXONWQPCLNCIJ
a=23, b=20: ZTYDYSZQFWFAMTZASCWOAJMGXWFZYLUWLRS
a=23, b=21: ICHMHBIZOFOJVCIJBLFXJSVPGFOIHUDFUAB
a=23, b=22: RLQVQKRIXOXSELRSKUOGSBEYPOXRQDMODJK
a=23, b=23: AUZEZTARGXGBNUABTDXPBKNHYXGAZMVXMST
a=23, b=24: JDINICJAPGPKWDJKCMGYKTWQHGPJIVEGVBC
a=23, b=25: SMRWRLSJYPYTFMSTLVPHTCFZQPYSRENPEKL
a=25, b=0: DLAPAIDCVUVGQLDGIMUWGHQYXUVDANOUNFI
a=25, b=1: EMBQBJEDWVWHRMEHJNVXHIRZYVWEBOPVOGJ
a=25, b=2: FNCRCKFEXWXISNFIKOWYIJSAZWXFCPQWPHK
a=25, b=3: GODSDLGFYXYJTOGJLPXZJKTBAXYGDQRXQIL
a=25, b=4: HPETEMHGZYZKUPHKMQYAKLUCBYZHERSYRJM
a=25, b=5: IQFUFNIHAZALVQILNRZBLMVDCZAIFSTZSKN
a=25, b=6: JRGVGOJIBABMWRJMOSACMNWEDABJGTUATLO
a=25, b=7: KSHWHPKJCBCNXSKNPTBDNOXFEBCKHUVBUMP
a=25, b=8: LTIXIQLKDCDOYTLOQUCEOPYGFCDLIVWCVNQ
a=25, b=9: MUJYJRMLEDEPZUMPRVDFPQZHGDEMJWXDWOR
a=25, b=10: NVKZKSNMFEFQAVNQSWEGQRAIHEFNKXYEXPS
a=25, b=11: OWLALTONGFGRBWORTXFHRSBJIFGOLYZFYQT
a=25, b=12: PXMBMUPOHGHSCXPSUYGISTCKJGHPMZAGZRU
a=25, b=13: QYNCNVQPIHITDYQTVZHJTUDLKHIQNABHASV
a=25, b=14: RZODOWRQJIJUEZRUWAIKUVEMLIJROBCIBTW
a=25, b=15: SAPEPXSRKJKVFASVXBJLVWFNMJKSPCDJCUX
a=25, b=16: TBQFQYTSLKLWGBTWYCKMWXGONKLTQDEKDVY
a=25, b=17: UCRGRZUTMLMXHCUXZDLNXYHPOLMUREFLEWZ
a=25, b=18: VDSHSAVUNMNYIDVYAEMOYZIQPMNVSFGMFXA
a=25, b=19: WETITBWVONOZJEWZBFNPZAJRQNOWTGHNGYB
```

```
a=25, b=19: WETITBWVONOZJEWZBFNPZAJRQNOWTGHNGYB
a=25, b=20: XFUJUCXWPOPAKFXACGOQABKSROPXUHIOHZC
a=25, b=21: YGVKVDYXQPQBLGYBDHPRBCLTSPQYVIJPIAD
a=25, b=22: ZHWLWEZYRQRCMHZCEIQSCDMUTQRZWJKQJBE
a=25, b=23: AIXMXFAZSRSDNIADFJRTDENVURSAXKLRKCF
a=25, b=24: BJYNYGBATSTEOJBEGKSUEFOWVSTBYLMSLDG
a=25, b=25: CKZOZHCBUTUFPKCFHLTVFGPXWTUCZMNTMEH

┌──(kali㉿kali)-[~/labPractice]
```

3. Encrypt the plain text using Rail Fence cipher

```python
def rail_fence_encrypt(plaintext, num_rails):
    """Encrypt the plaintext using Rail Fence cipher with a given number of rails."""
    # Remove spaces from plaintext and convert to uppercase
    plaintext = plaintext.replace(" ", "").upper()

    # Create a matrix to store the zigzag pattern
    rails = [['' for _ in range(len(plaintext))] for _ in range(num_rails)]

    # Variables to track the current position in the zigzag pattern
    direction = 1  # 1 means moving down, -1 means moving up
    row = 0

    # Fill the matrix in a zigzag manner
    for char in plaintext:
        rails[row].append(char)
        row += direction

        # Change direction if we hit the top or bottom rail
        if row == 0 or row == num_rails - 1:
            direction *= -1

    # Read off the matrix row by row
    ciphertext = ''.join(''.join(rail) for rail in rails)

    return ciphertext

# Plaintext to be encrypted
plaintext = "ENCYCLOPAEDIA"
num_rails = 3

# Perform Rail Fence cipher encryption
ciphertext = rail_fence_encrypt(plaintext, num_rails)
print(f"Ciphertext: {ciphertext}")
```

```
┌──(kali㉿kali)-[~/labPractice]
└─$ python lab5.py
Ciphertext: ECAANYLPEICOD
```

4. Decrypt the cipher using Rail Fence

AAIUJ SIHBE KTEAO TEADE SNUTF EAEMR TAHSA

RHROA YHNFO AITTE EHCBO FVCAT RNMNS NUTFE

RASHL WFHLN HIUJS IHTKS OEHNI FISAE FNTIG

RMRSO LSTIS OKIEH PEOE

```python
def rail_fence_decrypt(ciphertext, num_rails):
    """Decrypt the ciphertext using Rail Fence cipher with a given number of rails."""
    # Clean the ciphertext
    ciphertext = ''.join(ciphertext.split()).upper()
    n = len(ciphertext)

    # Create a matrix to reconstruct the zigzag pattern
    rails = [['' for _ in range(n)] for _ in range(num_rails)]

    # Initialize variables for zigzag traversal
    direction = 1  # 1 means moving down, -1 means moving up
    row = 0
    index = 0

    # Determine the positions of characters in the zigzag pattern
    positions = []
    for _ in range(n):
        positions.append((row, index))
        index += 1
        row += direction

        if row == 0 or row == num_rails - 1:
            direction *= -1

    # Fill the rails with the ciphertext characters
    rail_index = 0
    for r in range(num_rails):
        for c in range(n):
            if (r, c) in positions:
                rails[r][c] = ciphertext[rail_index]
```

```python
    # Fill the rails with the ciphertext characters
    rail_index = 0
    for r in range(num_rails):
        for c in range(n):
            if (r, c) in positions:
                rails[r][c] = ciphertext[rail_index]
                rail_index += 1

    # Read the rails in zigzag pattern to get the plaintext
    plaintext = ""
    row = 0
    direction = 1
    for _ in range(n):
        plaintext += rails[row][positions.index((row, _))]
        row += direction

        if row == 0 or row == num_rails - 1:
            direction *= -1

    return plaintext

# Ciphertext to be decrypted
ciphertext = """AAIUJ SIHBE KTEAO TEADE SNUTF EAEMR TAHSA
RHROA YHNFO AITTE EHCBO FVCAT RNMNS NUTFE
RASHL WFHLN HIUJS IHTKS OEHNI FISAE FNTIG
RMRSO LSTIS OKIEH PEOE"""

# Specify the number of rails used during encryption
num_rails = 3

# Perform Rail Fence cipher decryption
plaintext = rail_fence_decrypt(ciphertext, num_rails)
print(f"Plaintext: {plaintext}")
```

```
┌──(kali㉿kali)-[~/labPractice]
└─$ python lab5.py
Plaintext: AANHASIAIRFHURIOJASYSHANIFEOHAFIBTNTEETEKHICTBGOEFRVACMAOTRRTNSMENOSANLUDTSFEETRSAISNHSLUWOFTHKLFNIHEIEUAJHSEIPHMTEKRSOOTEEH
```

==========

Never stop trying

Never stop believe

Never give up

Your day will come.

You only fail when you stop trying.