

Quiz 01 - Practice

COMP 110: Introduction to Programming
Spring 2025

February 7, 2024

Name: Solutions

9-digit PID: _____

Do not begin until given permission.

Honor Code: I have neither given nor received any unauthorized aid on this quiz.

Signed: _____

Question 1: Multiple Choice Completely fill in the bubble next to your answer using a pencil. Each question should have exactly one filled-in bubble.

- 1.1. The following string is an example of a formatted string literal (f-string):

1 `"{1 + 1}"`

☐ True

☒ False

no f at the beginning!

- 1.2. What is the printed output of the following `print` function call?

1 `print(f"C{'OM'}P{100 + 10}")`

☐ `fCOMP10010`

☒ `COMP110`

☐ `C'OM'P100 + 10`

☐ Error: Invalid Syntax

- 1.3. What is the *type* and *evaluation* of this expression in Python?

1 `"ABCD" == "ABcd"`

☐ `bool, True`

☒ `bool, False`

Python is case-sensitive!

- 1.4. What is the primary difference between keyword arguments and positional arguments in Python?

☐ Keyword arguments must always be passed, while positional arguments are optional.

☒ Positional arguments are passed based on their position in the function call, while keyword arguments are explicitly named.

☐ Keyword arguments can only be used in built-in functions, while positional arguments can be used in both built-in and user-defined functions.

☐ Positional arguments must always come after keyword arguments in a function call.

- 1.5. Which operator has the highest precedence in an expression?

☐ `or`

☐ `>`

☒ `+`

☐ `and`

☐ `not`

- 1.6. Which of the following statements correctly describes the behavior of the `and`, `or`, and `not` operators in Python?

☐ The `and` operator returns True if at least one operand is True.

☐ The `or` operator returns True only if both operands are True.

☒ The `not` operator inverts the boolean value of an expression.

☐ The `and`, `or`, and `not` operators can only be used with boolean values.

- 1.7. What is the evaluation of the following expression:

1 `1 > 0 or 6 > 8`

True

☐ False

☒ True

- 1.8. What is the evaluation of the following expression:

1 `"A" == "B" and "B" == "C"`

☒ False

☐ True

1.9. What is the evaluation of the following Python expression?

```
1 not True or True
```

- ☐ False
- ☒ True
- ☐ Error

1.10. Which of the following are required in a recursive function that does not infinitely recur?

- ☐ A base case without a recursive function call
- ☐ Recursive case that progresses toward the base case
- ☐ Arguments changing in the recursive case
- ☒ All of the above

1.11. Which of the following is a valid function call to the following function signature?

```
1 def ex(x: int, y: int=0) -> int:  
2 ...
```

- ☐ A. `ex()`
- ☐ B. `ex(1)`
- ☐ C. `ex(1, 2)`
- ☒ B and C
- ☐ A, B, and C
- ☐ None of the above

default parameter

1.12. What type of error occurs when a function keeps calling itself, indefinitely?

- ☐ `NameError`
- ☐ `IndexError`
- ☒ `RecursionError`
- ☐ `SyntaxError`
- ☐ `NeverendingError`

1.13. What will the following Python expression evaluate to?

```
1 1 + True
```

- ☐ True
- ☒ 2
- ☐ 1
- ☐ False

1.14. Consider the following function declaration:

```
1 def ex(x: int, y: int=0) -> int:  
2 ...
```

Which of the following are valid ways of calling the function?

- ☐ A. `ex(x=1, y=2)` *keyword arguments*
- ☐ B. `ex(x=1)`
- ☐ C. `ex(1, 2)` *positional arguments*
- ☐ A and B
- ☒ A, B, and C
- ☐ None of the above

1.15. Consider the following code. What is the problem with it?

```
1 def charli(x: int) -> int:  
2     if x <= 0:  
3         return 1  
4     return x + charli(x)
```

- ☐ `RecursionError`; line 4 should be `return x * charli(x)`
- ☒ `RecursionError`; line 4 should be `return x + charli(x - 1)`
- ☐ `RecursionError`; line 4 should be `return x + charli(x + 1)`
- ☐ Nothing.

the recursive case has to progress toward the base case!

Question 2: Respond to the following questions. Write a function call, if any, to yield the correct return value.

Consider the following code listing:

```
1 def eight_ball(choice: int) -> str:
2     """Returns an 8-ball response."""
3     if choice <= 0:
4         return "Unlikely."
5     else:
6         if choice > 0:
7             return "It is certain."
8         else:
9             return "Ask again later."
```

2.1. Write a function call expression to the `eight_ball` function that evaluates to "It is certain."

Solution: `eight_ball(1)` or `eight_ball(choice=1)` or any argument value greater than 1

2.2. Write a function call expression to the `eight_ball` function that evaluates to "Unlikely."

Solution: `eight_ball(0)` or `eight_ball(choice=0)` or any argument value less than 0

2.3. Write a function call expression to the `eight_ball` function that evaluates to "Ask again later."

Solution: This code is unreachable and no function call can be made, as written, to result in "Ask again later."

2.4. Rewrite lines 3-9 of the code listing to eliminate any unreachable code and the nested if-else statement.

```
if choice <= 0:
    return "Unlikely."
else:
    return "It is certain."
```

Question 3: Respond to the following questions.

3.1. What value and type does the following expression evaluate to: `3 + 4 == 6`

Solution: False, bool

3.2. What value and type does the following expression evaluate to?

```
1 ((True and False) or (False or True)) != False
```

Solution: True

Question 4: Memory Diagram Trace a memory diagram of the following code listing and then answer the sub questions. You do not need to diagram the sub questions.

```

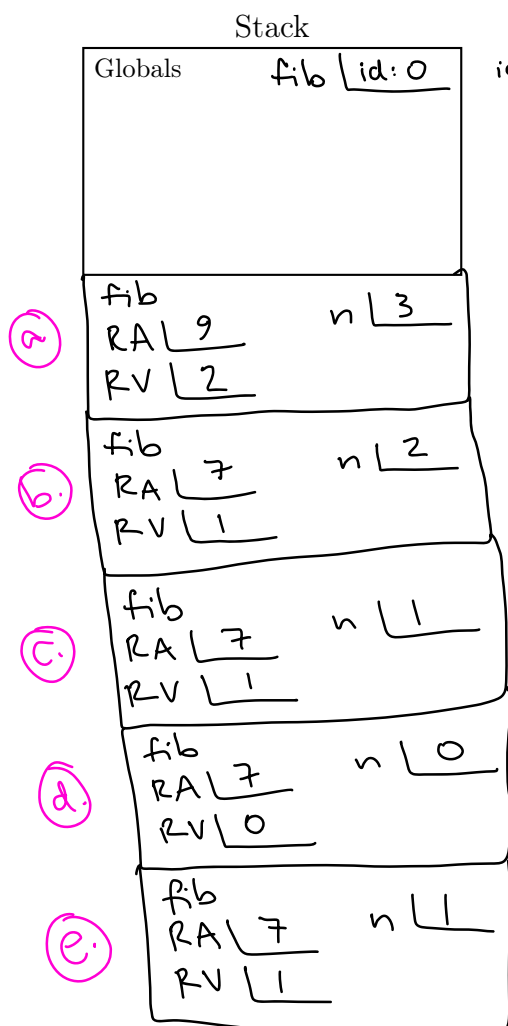
1 def fib(n: int) -> int:
2     """Compute the fibonacci of n"""
3     print(f"fib({n})")
4     if n == 0 or n == 1:
5         return n
6     else:
7         return fib(n - 1) + fib(n - 2)
8
9 print(fib(3))

```

← ignore that

Output

Solution: fib(3) // fib(2) // fib(1) // fib(0) // fib(1) // 2



Heap
id: 0 fn lines 1-7

* Not needed in diagram, but here is a visualization of the control flow:

```

graph LR
    a((a)) --> b((b))
    b --> c((c))
    c --> d((d))
    d --> e((e))
    e --> f((f))
    f --> g((g))
    g --> h((h))
    h --> i((i))
    i --> j((j))
    j --> k((k))
    k --> l((l))
    l --> m((m))
    m --> n((n))
    n --> o((o))
    o --> p((p))
    p --> q((q))
    q --> r((r))
    r --> s((s))
    s --> t((t))
    t --> u((u))
    u --> v((v))
    v --> w((w))
    w --> x((x))
    x --> y((y))
    y --> z((z))
    z --> aa((aa))
    aa --> ab((ab))
    ab --> ac((ac))
    ac --> ad((ad))
    ad --> ae((ae))
    ae --> af((af))
    af --> ag((ag))
    ag --> ah((ah))
    ah --> ai((ai))
    ai --> aj((aj))
    aj --> ak((ak))
    ak --> al((al))
    al --> am((am))
    am --> an((an))
    an --> ao((ao))
    ao --> ap((ap))
    ap --> aq((aq))
    aq --> ar((ar))
    ar --> as((as))
    as --> at((at))
    at --> au((au))
    au --> av((av))
    av --> aw((aw))
    aw --> ax((ax))
    ax --> ay((ay))
    ay --> az((az))
    az --> ba((ba))
    ba --> bb((bb))
    bb --> bc((bc))
    bc --> bd((bd))
    bd --> be((be))
    be --> bf((bf))
    bf --> bg((bg))
    bg --> bh((bh))
    bh --> bi((bi))
    bi --> bj((bj))
    bj --> bk((bk))
    bk --> bl((bl))
    bl --> bm((bm))
    bm --> bn((bn))
    bn --> bo((bo))
    bo --> bp((bp))
    bp --> bq((bq))
    bq --> br((br))
    br --> bs((bs))
    bs --> bt((bt))
    bt --> bu((bu))
    bu --> bv((bv))
    bv --> bw((bw))
    bw --> bx((bx))
    bx --> by((by))
    by --> bz((bz))
    bz --> ca((ca))
    ca --> cb((cb))
    cb --> cc((cc))
    cc --> cd((cd))
    cd --> ce((ce))
    ce --> cf((cf))
    cf --> cg((cg))
    cg --> ch((ch))
    ch --> ci((ci))
    ci --> cj((cj))
    cj --> ck((ck))
    ck --> cl((cl))
    cl --> cm((cm))
    cm --> cn((cn))
    cn --> co((co))
    co --> cp((cp))
    cp --> cq((cq))
    cq --> cr((cr))
    cr --> cs((cs))
    cs --> ct((ct))
    ct --> cu((cu))
    cu --> cv((cv))
    cv --> cw((cw))
    cw --> cx((cx))
    cx --> cy((cy))
    cy --> cz((cz))
    cz --> da((da))
    da --> db((db))
    db --> dc((dc))
    dc --> dd((dd))
    dd --> de((de))
    de --> df((df))
    df --> dg((dg))
    dg --> dh((dh))
    dh --> di((di))
    di --> dj((dj))
    dj --> dk((dk))
    dk --> dl((dl))
    dl --> dm((dm))
    dm --> dn((dn))
    dn --> do((do))
    do --> dp((dp))
    dp --> dq((dq))
    dq --> dr((dr))
    dr --> ds((ds))
    ds --> dt((dt))
    dt --> du((du))
    du --> dv((dv))
    dv --> dw((dw))
    dw --> dx((dx))
    dx --> dy((dy))
    dy --> dz((dz))
    dz --> ea((ea))
    ea --> eb((eb))
    eb --> ec((ec))
    ec --> ed((ed))
    ed --> ee((ee))
    ee --> ef((ef))
    ef --> eg((eg))
    eg --> eh((eh))
    eh --> ei((ei))
    ei --> ej((ej))
    ej --> ek((ek))
    ek --> el((el))
    el --> em((em))
    em --> en((en))
    en --> eo((eo))
    eo --> ep((ep))
    ep --> eq((eq))
    eq --> er((er))
    er --> es((es))
    es --> et((et))
    et --> eu((eu))
    eu --> ev((ev))
    ev --> ew((ew))
    ew --> ex((ex))
    ex --> ey((ey))
    ey --> ez((ez))
    ez --> fa((fa))
    fa --> fb((fb))
    fb --> fc((fc))
    fc --> fd((fd))
    fd --> fe((fe))
    fe --> ff((ff))
    ff --> fg((fg))
    fg --> fh((fh))
    fh --> fi((fi))
    fi --> fj((fj))
    fj --> fk((fk))
    fk --> fl((fl))
    fl --> fm((fm))
    fm --> fn((fn))
    fn --> fo((fo))
    fo --> fp((fp))
    fp --> fq((fq))
    fq --> fr((fr))
    fr --> fs((fs))
    fs --> ft((ft))
    ft --> fu((fu))
    fu --> fv((fv))
    fv --> fw((fw))
    fw --> fx((fx))
    fx --> fy((fy))
    fy --> fz((fz))
    fz --> ga((ga))
    ga --> gb((gb))
    gb --> gc((gc))
    gc --> gd((gd))
    gd --> ge((ge))
    ge --> gf((gf))
    gf --> gg((gg))
    gg --> gh((gh))
    gh --> gi((gi))
    gi --> gj((gj))
    gj --> gk((gk))
    gk --> gl((gl))
    gl --> gm((gm))
    gm --> gn((gn))
    gn --> go((go))
    go --> gp((gp))
    gp --> gq((gq))
    gq --> gr((gr))
    gr --> gs((gs))
    gs --> gt((gt))
    gt --> gu((gu))
    gu --> gv((gv))
    gv --> gw((gw))
    gw --> gx((gx))
    gx --> gy((gy))
    gy --> gz((gz))
    gz --> ha((ha))
    ha --> hb((hb))
    hb --> hc((hc))
    hc --> hd((hd))
    hd --> he((he))
    he --> hf((hf))
    hf --> hg((hg))
    hg --> hi((hi))
    hi --> hj((hj))
    hj --> hk((hk))
    hk --> hl((hl))
    hl --> hm((hm))
    hm --> hn((hn))
    hn --> ho((ho))
    ho --> hp((hp))
    hp --> hq((hq))
    hq --> hr((hr))
    hr --> hs((hs))
    hs --> ht((ht))
    ht --> hu((hu))
    hu --> hv((hv))
    hv --> hw((hw))
    hw --> hx((hx))
    hx --> hy((hy))
    hy --> hz((hz))
    hz --> ia((ia))
    ia --> ib((ib))
    ib --> ic((ic))
    ic --> id((id))
    id --> ie((ie))
    ie --> if((if))
    if --> ig((ig))
    ig --> ih((ih))
    ih --> ii((ii))
    ii --> ij((ij))
    ij --> ik((ik))
    ik --> il((il))
    il --> im((im))
    im --> in((in))
    in --> io((io))
    io --> ip((ip))
    ip --> iq((iq))
    iq --> ir((ir))
    ir --> is((is))
    is --> it((it))
    it --> iu((iu))
    iu --> iv((iv))
    iv --> iw((iw))
    iw --> ix((ix))
    ix --> iy((iy))
    iy --> iz((iz))
    iz --> ja((ja))
    ja --> jb((jb))
    jb --> jc((jc))
    jc --> jd((jd))
    jd --> je((je))
    je --> jf((jf))
    jf --> jg((jg))
    jg --> jh((jh))
    jh --> ji((ji))
    ji --> jj((jj))
    jj --> jk((jk))
    jk --> jl((jl))
    jl --> jm((jm))
    jm --> jn((jn))
    jn --> jo((jo))
    jo --> jp((jp))
    jp --> jq((jq))
    jq --> jr((jr))
    jr --> js((js))
    js --> jt((jt))
    jt --> ju((ju))
    ju --> jv((jv))
    jv --> jw((jw))
    jw --> jx((jx))
    jx --> jy((jy))
    jy --> jz((jz))
    jz --> ka((ka))
    ka --> kb((kb))
    kb --> kc((kc))
    kc --> kd((kd))
    kd --> ke((ke))
    ke --> kf((kf))
    kf --> kg((kg))
    kg --> kh((kh))
    kh --> ki((ki))
    ki --> kj((kj))
    kj --> kk((kk))
    kk --> kl((kl))
    kl --> km((km))
    km --> kn((kn))
    kn --> ko((ko))
    ko --> kp((kp))
    kp --> kq((kq))
    kq --> kr((kr))
    kr --> ks((ks))
    ks --> kt((kt))
    kt --> ku((ku))
    ku --> kv((kv))
    kv --> kw((kw))
    kw --> kx((kx))
    kx --> ky((ky))
    ky --> kz((kz))
    kz --> la((la))
    la --> lb((lb))
    lb --> lc((lc))
    lc --> ld((ld))
    ld --> le((le))
    le --> lf((lf))
    lf --> lg((lg))
    lg --> lh((lh))
    lh --> li((li))
    li --> lj((lj))
    lj --> lk((lk))
    lk --> ll((ll))
    ll --> lm((lm))
    lm --> ln((ln))
    ln --> lo((lo))
    lo --> lp((lp))
    lp --> lq((lq))
    lq --> lr((lr))
    lr --> ls((ls))
    ls --> lt((lt))
    lt --> lu((lu))
    lu --> lv((lv))
    lv --> lw((lw))
    lw --> lx((lx))
    lx --> ly((ly))
    ly --> lz((lz))
    lz --> ma((ma))
    ma --> mb((mb))
    mb --> mc((mc))
    mc --> md((md))
    md --> me((me))
    me --> mf((mf))
    mf --> mg((mg))
    mg --> mh((mh))
    mh --> mi((mi))
    mi --> mj((mj))
    mj --> mk((mk))
    mk --> ml((ml))
    ml --> mn((mn))
    mn --> mo((mo))
    mo --> mp((mp))
    mp --> mq((mq))
    mq --> mr((mr))
    mr --> ms((ms))
    ms --> mt((mt))
    mt --> mu((mu))
    mu --> mv((mv))
    mv --> mw((mw))
    mw --> mx((mx))
    mx --> my((my))
    my --> mz((mz))
    mz --> na((na))
    na --> nb((nb))
    nb --> nc((nc))
    nc --> nd((nd))
    nd --> ne((ne))
    ne --> nf((nf))
    nf --> ng((ng))
    ng --> nh((nh))
    nh --> ni((ni))
    ni --> nj((nj))
    nj --> nk((nk))
    nk --> nl((nl))
    nl --> nm((nm))
    nm --> no((no))
    no --> np((np))
    np --> nq((nq))
    nq --> nr((nr))
    nr --> ns((ns))
    ns --> nt((nt))
    nt --> nu((nu))
    nu --> nv((nv))
    nv --> nw((nw))
    nw --> nx((nx))
    nx --> ny((ny))
    ny --> nz((nz))
    nz --> oa((oa))
    oa --> ob((ob))
    ob --> oc((oc))
    oc --> od((od))
    od --> oe((oe))
    oe --> of((of))
    of --> og((og))
    og --> oh((oh))
    oh --> oi((oi))
    oi --> oj((oj))
    oj --> ok((ok))
    ok --> ol((ol))
    ol --> om((om))
    om --> on((on))
    on --> oo((oo))
    oo --> op((op))
    op --> oq((oq))
    oq --> or((or))
    or --> os((os))
    os --> ot((ot))
    ot --> ou((ou))
    ou --> ov((ov))
    ov --> ow((ow))
    ow --> ox((ox))
    ox --> oy((oy))
    oy --> oz((oz))
    oz --> pa((pa))
    pa --> pb((pb))
    pb --> pc((pc))
    pc --> pd((pd))
    pd --> pe((pe))
    pe --> pf((pf))
    pf --> pg((pg))
    pg --> ph((ph))
    ph --> pi((pi))
    pi --> pj((pj))
    pj --> pk((pk))
    pk --> pl((pl))
    pl --> pm((pm))
    pm --> pn((pn))
    pn --> po((po))
    po --> pp((pp))
    pp --> pq((pq))
    pq --> pr((pr))
    pr --> ps((ps))
    ps --> pt((pt))
    pt --> pu((pu))
    pu --> pv((pv))
    pv --> pw((pw))
    pw --> px((px))
    px --> py((py))
    py --> pz((pz))
    pz --> qa((qa))
    qa --> qb((qb))
    qb --> qc((qc))
    qc --> qd((qd))
    qd --> qe((qe))
    qe --> qf((qf))
    qf --> qg((qg))
    qg --> qh((qh))
    qh --> qi((qi))
    qi --> qj((qj))
    qj --> qk((qk))
    qk --> ql((ql))
    ql --> qm((qm))
    qm --> qn((qn))
    qn --> qo((qo))
    qo --> qp((qp))
    qp --> qq((qq))
    qq --> qr((qr))
    qr --> qs((qs))
    qs --> qt((qt))
    qt --> qu((qu))
    qu --> qv((qv))
    qv --> qw((qw))
    qw --> qx((qx))
    qx --> qy((qy))
    qy --> qz((qz))
    qz --> ra((ra))
    ra --> rb((rb))
    rb --> rc((rc))
    rc --> rd((rd))
    rd --> re((re))
    re --> rf((rf))
    rf --> rg((rg))
    rg --> rh((rh))
    rh --> ri((ri))
    ri --> rj((rj))
    rj --> rk((rk))
    rk --> rl((rl))
    rl --> rm((rm))
    rm --> rn((rn))
    rn --> ro((ro))
    ro --> rp((rp))
    rp --> rq((rq))
    rq --> rr((rr))
    rr --> rs((rs))
    rs --> rt((rt))
    rt --> ru((ru))
    ru --> rv((rv))
    rv --> rw((rw))
    rw --> rx((rx))
    rx --> ry((ry))
    ry --> rz((rz))
    rz --> sa((sa))
    sa --> sb((sb))
    sb --> sc((sc))
    sc --> sd((sd))
    sd --> se((se))
    se --> sf((sf))
    sf --> sg((sg))
    sg --> sh((sh))
    sh --> si((si))
    si --> sj((sj))
    sj --> sk((sk))
    sk --> sl((sl))
    sl --> sm((sm))
    sm --> sn((sn))
    sn --> so((so))
    so --> sp((sp))
    sp --> sq((sq))
    sq --> sr((sr))
    sr --> ss((ss))
    ss --> st((st))
    st --> su((su))
    su --> sv((sv))
    sv --> sw((sw))
    sw --> sx((sx))
    sx --> sy((sy))
    sy --> sz((sz))
    sz --> ta((ta))
    ta --> tb((tb))
    tb --> tc((tc))
    tc --> td((td))
    td --> te((te))
    te --> tf((tf))
    tf --> tg((tg))
    tg --> th((th))
    th --> ti((ti))
    ti --> tj((tj))
    tj --> tk((tk))
    tk --> tl((tl))
    tl --> tm((tm))
    tm --> tn((tn))
    tn --> to((to))
    to --> tp((tp))
    tp --> tq((tq))
    tq --> tr((tr))
    tr --> ts((ts))
    ts --> tt((tt))
    tt --> tu((tu))
    tu --> tv((tv))
    tv --> tw((tw))
    tw --> tx((tx))
    tx --> ty((ty))
    ty --> tz((tz))
    tz --> ua((ua))
    ua --> ub((ub))
    ub --> uc((uc))
    uc --> ud((ud))
    ud --> ue((ue))
    ue --> uf((uf))
    uf --> ug((ug))
    ug --> uh((uh))
    uh --> ui((ui))
    ui --> uj((uj))
    uj --> uk((uk))
    uk --> ul((ul))
    ul --> um((um))
    um --> un((un))
    un --> uo((uo))
    uo --> up((up))
    up --> uq((uq))
    uq --> ur((ur))
    ur --> us((us))
    us --> ut((ut))
    ut --> uu((uu))
    uu --> uv((uv))
    uv --> uw((uw))
    uw --> ux((ux))
    ux --> uy((uy))
    uy --> uz((uz))
    uz --> va((va))
    va --> vb((vb))
    vb --> vc((vc))
    vc --> vd((vd))
    vd --> ve((ve))
    ve --> vf((vf))
    vf --> vg((vg))
    vg --> vh((vh))
    vh --> vi((vi))
    vi --> vj((vj))
    vj --> vk((vk))
    vk --> vl((vl))
    vl --> vm((vm))
    vm --> vn((vn))
    vn --> vo((vo))
    vo --> vp((vp))
    vp --> vq((vq))
    vq --> vr((vr))
    vr --> vs((vs))
    vs --> vt((vt))
    vt --> vu((vu))
    vu --> vv((vv))
    vv --> vw((vw))
    vw --> vx((vx))
    vx --> vy((vy))
    vy --> vz((vz))
    vz --> wa((wa))
    wa --> wb((wb))
    wb --> wc((wc))
    wc --> wd((wd))
    wd --> we((we))
    we --> wf((wf))
    wf --> wg((wg))
    wg --> wh((wh))
    wh --> wi((wi))
    wi --> wj((wj))
    wj --> wk((wk))
    wk --> wl((wl))
    wl --> wm((wm))
    wm --> wn((wn))
    wn --> wo((wo))
    wo --> wp((wp))
    wp --> wq((wq))
    wq --> wr((wr))
    wr --> ws((ws))
    ws --> wt((wt))
    wt --> wu((wu))
    wu --> wv((wv))
    wv --> ww((ww))
    ww --> wx((wx))
    wx --> wy((wy))
    wy --> wz((wz))
    wz --> xa((xa))
    xa --> xb((xb))
    xb --> xc((xc))
    xc --> xd((xd))
    xd --> xe((xe))
    xe --> xf((xf))
    xf --> xg((xg))
    xg --> xh((xh))
    xh --> xi((xi))
    xi --> xj((xj))
    xj --> xk((xk))
    xk --> xl((xl))
    xl --> xm((xm))
    xm --> xn((xn))
    xn --> xo((xo))
    xo --> xp((xp))
    xp --> xq((xq))
    xq --> xr((xr))
    xr --> xs((xs))
    xs --> xt((xt))
    xt --> xu((xu))
    xu --> xv((xv))
    xv --> xw((xw))
    xw --> xx((xx))
    xx --> xy((xy))
    xy --> xz((xz))
    xz --> ya((ya))
    ya --> yb((yb))
    yb --> yc((yc))
    yc --> yd((yd))
    yd --> ye((ye))
    ye --> yf((yf))
    yf --> yg((yg))
    yg --> yh((yh))
    yh --> yi((yi))
    yi --> yj((yj))
    yj --> yk((yk))
    yk --> yl((yl))
    yl --> ym((ym))
    ym --> yn((yn))
    yn --> yo((yo))
    yo --> yp((yp))
    yp --> yq((yq))
    yq --> yr((yr))
    yr --> ys((ys))
    ys --> yt((yt))
    yt --> yu((yu))
    yu --> yv((yv))
    yv --> yw((yw))
    yw --> yx((yx))
    yx --> yy((yy))
    yy --> yz((yz))
    yz --> za((za))
    za --> zb((zb))
    zb --> zc((zc))
    zc --> zd((zd))
    zd --> ze((ze))
    ze --> zf((zf))
    zf --> zg((zg))
    zg --> zh((zh))
    zh --> zi((zi))
    zi --> zj((zj))
    zj --> zk((zk))
    zk --> zl((zl))
    zl --> zm((zm))
    zm --> zn((zn))
    zn --> zo((zo))
    zo --> zp((zp))
    zp --> zq((zq))
    zq --> zr((zr))
    zr --> zs((zs))
    zs --> zt((zt))
    zt --> zu((zu))
    zu --> zv((zv))
    zv --> zw((zw))
    zw --> zx((zx))
    zx --> zy((zy))
    zy --> zz((zz))
    zz --> AA((AA))
    AA --> AB((AB))
    AB --> AC((AC))
    AC --> AD((AD))
    AD --> AE((AE))
    AE --> AF((AF))
    AF --> AG((AG))
    AG --> AH((AH))
    AH --> AI((AI))
    AI --> AJ((AJ))
    AJ --> AK((AK))
    AK --> AL((AL))
    AL --> AM((AM))
    AM --> AN((AN))
    AN --> AO((AO))
    AO --> AP((AP))
    AP --> AQ((AQ))
    AQ --> AR((AR))
    AR --> AS((AS))
    AS --> AT((AT))
    AT --> AU((AU))
    AU --> AV((AV))
    AV --> AW((AW))
    AW --> AX((AX))
    AX --> AY((AY))
    AY --> AZ((AZ))
    AZ --> BA((BA))
    BA --> BB((BB))
    BB --> BC((BC))
    BC --> BD((BD))
    BD --> BE((BE))
    BE --> BF((BF))
    BF --> BG((BG))
    BG --> BH((BH))
    BH --> BI((BI))
    BI --> BJ((BJ))
    BJ --> BK((BK))
    BK --> BL((BL))
    BL --> BM((BM))
    BM --> BN((BN))
    BN --> BO((BO))
    BO --> BP((BP))
    BP --> BQ((BQ))
    BQ --> BR((BR))
    BR --> BS((BS))
    BS --> BT((BT))
    BT --> BU((BU))
    BU --> BV((BV))
    BV --> BW((BW))
    BW --> BX((BX))
    BX --> BY((BY))
    BY --> BZ((BZ))
    BZ --> CA((CA))
    CA --> CB((CB))
    CB --> CC((CC))
    CC --> CD((CD))
    CD --> CE((CE))
    CE --> CF((CF))
    CF --> CG((CG))
    CG --> CH((CH))
    CH --> CI((CI))
    CI --> CJ((CJ))
    CJ --> CK((CK))
    CK --> CL((CL))
    CL --> CM((CM))
    CM --> CN((CN))
    CN --> CO((CO))
    CO --> CP((CP))
    CP --> CQ((CQ))
    CQ --> CR((CR))
    CR --> CS((CS))
    CS --> CT((CT))
    CT --> CU((CU))
    CU --> CV((CV))
    CV --> CW((CW))
    CW --> CX((CX))
    CX --> CY((CY))
    CY --> CZ((CZ))
    CZ --> DA((DA))
    DA --> DB((DB))
    DB --> DC((DC))
    DC --> DE((DE))
    DE --> DF((DF))
    DF --> DG((DG))
    DG --> DH((DH))
    DH --> DI((DI))
    DI --> DJ((DJ))
    DJ --> DK((DK))
    DK --> DL((DL))
    DL --> DM((DM))
    DM --> DN((DN))
    DN --> DO((DO))
    DO --> DP((DP))
    DP --> DQ((DQ))
    DQ --> DR((DR))
    DR --> DS((DS))
    DS --> DT((DT))
    DT --> DU((DU))
    DU --> DV((DV))
    DV --> DW((DW))
    DW --> DX((DX))
    DX --> DY((DY))
    DY --> DZ((DZ))
    DZ --> EA((EA))
    EA --> EB((EB))
    EB --> EC((EC))
    EC --> ED((ED))
    ED --> EE((EE))
    EE --> EF((EF))
    EF --> EG((EG))
    EG --> EH((EH))
    EH --> EI((EI))
    EI --> EJ((EJ))
    EJ --> EK((EK))
    EK --> EL((EL))
    EL --> EM((EM))
    EM --> EN((EN))
    EN --> EO((EO))
    EO --> EP((EP))
    EP --> EQ((EQ))
    EQ --> ER((ER))
    ER --> ES((ES))
    ES --> ET((ET))
    ET --> EU((EU))
    EU --> EV((EV))
    EV --> EW((EW))
    EW --> EX((EX))
    EX --> EY((EY))
    EY --> EZ((EZ))
    EZ --> FA((FA))
    FA --> FB((FB))
    FB --> FC((FC))
    FC --> FD((FD))
    FD --> FE((FE))
    FE --> FF((FF))
    FF --> FG((FG))
    FG --> FH((FH))
    FH --> FI((FI))
    FI --> FJ((FJ))
    FJ --> FK((FK))
    FK --> FL((FL))
    FL --> FM((FM))
    FM --> FN((FN))
    FN --> FO((FO))
    FO --> FP((FP))
    FP --> FQ((FQ))
    FQ --> FR((FR))
    FR --> FS((FS))
    FS --> FT((FT))
    FT --> FU((FU))
    FU --> FV((FV))
    FV --> FW((FW))
    FW --> FX((FX))
    FX --> FY((FY))
    FY --> FZ((FZ))
    FZ --> GA((GA))
    GA --> GB((GB))
    GB --> GC((GC))
    GC --> GD((GD))
    GD --> GE((GE))
    GE --> GF((GF))
    GF --> GG((GG))
    GG --> GH((GH))
    GH --> GI((GI))
    GI --> GJ((GJ))
    GJ --> GK((GK))
    GK --> GL((GL))
    GL --> GM((GM))
    GM --> GN((GN))
    GN --> GO((GO))
    GO --> GP((GP))
    GP --> GQ((GQ))
    GQ --> GR((GR))
    GR --> GS((GS))
    GS --> GT((GT))
    GT --> GU((GU))
    GU --> GV((GV))
    GV --> GW((GW))
    GW --> GX((GX))
    GX --> GY((GY))
    GY --> GZ((GZ))
    GZ --> HA((HA))
    HA --> HB((HB))
    HB --> HC((HC))
    HC --> HD((HD))
    HD --> HE((HE))
    HE --> HF((HF))
    HF --> HG((HG))
    HG --> HI((HI))
    HI --> HJ((HJ))
    HJ --> HK((HK))
    HK --> HL((HL))
    HL --> HM((HM))
    HM --> HN((HN))
    HN --> HO((HO))
    HO --> HP((HP))
    HP --> HQ((HQ))
    HQ --> HR((HR))
    HR --> HS((HS))
    HS --> HT((HT))
    HT --> HU((HU))
    HU --> HV((HV))
    HV --> HW((HW))
    HW --> HX((HX))
    HX --> HY((HY))
    HY --> HZ((HZ))
    HZ --> IA((IA))
    IA --> IB((IB))
    IB --> IC((IC))
    IC --> ID((ID))
    ID --> IE((IE))
    IE --> IF((IF))
    IF --> IG((IG))
    IG --> IH((IH))
    IH --> II((II))
    II --> IJ((IJ))
    IJ --> IK((IK))
    IK --> IL((IL))
    IL --> IM((IM))
    IM --> IN((IN))
    IN --> IO((IO))
    IO --> IP((IP))
    IP --> IQ((IQ))
    IQ --> IR((IR))
    IR --> IS((IS))
    IS --> IT((IT))
    IT --> IU((IU))
    IU --> IV((IV))
    IV --> IW((IW))
    IW --> IX((IX))
    IX --> IY((IY))
    IY --> IZ((IZ))
    IZ --> JA((JA))
    JA --> JB((JB))
    JB --> JC((JC))
    JC --> JD((JD))
    JD --> JE((JE))
    JE --> JF((JF))
    JF --> JG((JG))
    JG --> JH((JH))
    JH --> JI((JI))
    JI --> JJ((JJ))
    JJ --> JK((JK))
    JK --> JL((JL))
    JL --> JM((JM))
    JM --> JN((JN))
    JN --> JO((JO))
    JO --> JP((JP))
    JP --> JQ((JQ))
    JQ --> JR((JR))
    JR --> JS((JS))
    JS --> JT((JT))
    JT --> JU((JU))
    JU --> JV((JV))
    JV --> JW((JW))
    JW --> JX((JX))
    JX --> JY((JY))
    JY --> JZ((JZ))
    JZ --> KA((KA))
    KA --> KB((KB))
    KB --> KC((KC))
    KC --> KD((KD))
    KD --> KE((KE))
    KE --> KF((KF))
    KF --> KG((KG))
    KG --> KH((KH))
    KH --> KI((KI))
    KI --> KJ((KJ))
    KJ --> KL((KL))
    KL --> KM((KM))
    KM --> KN((KN))
    KN --> KO((KO))
    KO --> KP((KP))
    KP --> KQ((KQ))
    KQ --> KR((KR))
    KR --> KS((KS))
    KS --> KT((KT))
    KT --> KU((KU))
    KU --> KV((KV))
    KV --> KW((KW))
    KW --> KX((KX))
    KX --> KY((KY))
    KY --> KZ((KZ))
    KZ --> LA((LA))
    LA --> LB((LB))
    LB --> LC((LC))
    LC --> LD((LD))
    LD --> LE((LE))
    LE --> LF((LF))
    LF --> LG((LG))
    LG --> LH((LH))
    LH --> LI((LI))
    LI --> LJ((LJ))
    LJ --> LK((LK))
    LK --> LL((LL))
    LL --> LM((LM))
    LM --> LN((LN))
    LN --> LO((LO))
    LO --> LP((LP))
    LP --> LQ((LQ))
    LQ --> LR((LR))
    LR --> LS((LS))
    LS --> LT((LT))
    LT --> LU((LU))
    LU --> LV((LV))
    LV --> LW((LW))
    LW --> LX((LX))
    LX --> LY((LY))
    LY --> LZ((LZ))
    LZ --> MA((MA))
    MA --> MB((MB))
    MB --> MC((MC))
    MC --> MD((MD))
    MD --> ME((ME))
    ME --> MF((MF))
    MF --> MG((MG))
    MG --> MH((MH))
    MH --> MI((MI))
    MI --> MJ((MJ))
    MJ --> MK((MK))
    MK --> ML((ML))
    ML --> MN((MN))
    MN --> MO((MO))
    MO --> MP((MP))
    MP --> MQ((MQ))
    MQ --> MR((MR))
    MR --> MS((MS))
    MS --> MT((MT))
    MT --> MU((MU))
    MU --> MV((MV))
    MV --> MW((MW))
    MW --> MX((MX))
   
```

Question 5: Memory Diagram Trace a memory diagram of the following code listing ~~and~~ then answer the sub questions. You do not need to diagram the sub questions.

```

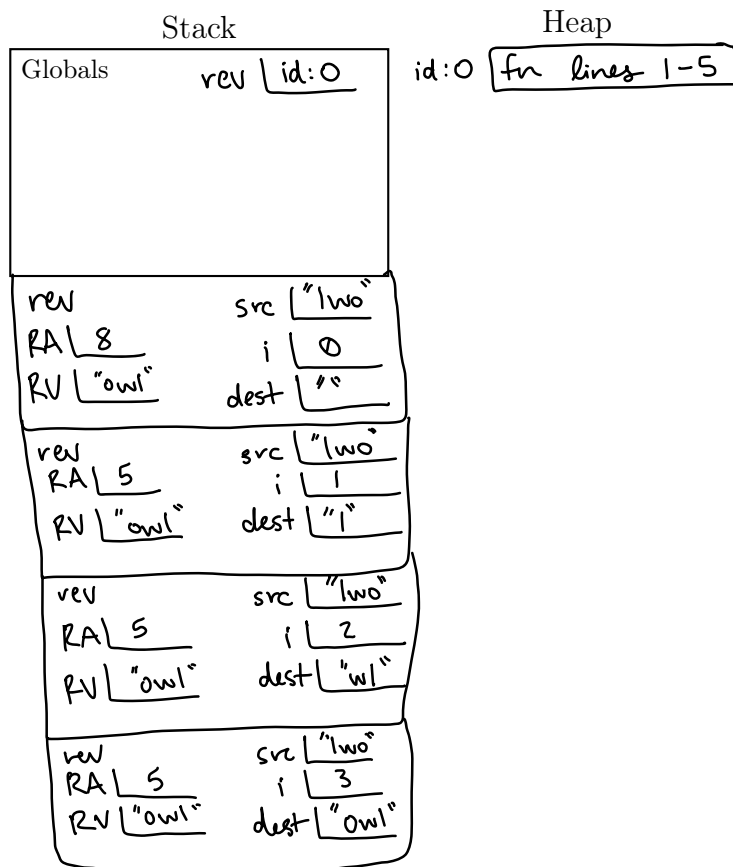
1 def rev(src: str, i: int, dest: str) -> str:
2     if i >= len(src):
3         return dest
4     else:
5         return rev(src=src, i=i + 1, dest=src[i] + dest)
6
7
8 print(rev(src="lwo", i=0, dest=""))

```

← ignore that

Output

owl



Question 6: Function Definition Writing Write a function definition that returns a different string, depending on the value of a given `int`. Your function definition should meet the following expectations:

- The function should be named `fizzbuzz`, have one `int` parameter named `n`, and return a `str`.
- If `n` is divisible by 3 and not 5, the function should return `"fizz"`.
- If `n` is divisible by 5 and not 3, the function should return `"buzz"`.
- If `n` is divisible by 3 AND 5, the function should return `"fizzbuzz"`.
- If `n` is not divisible by 3 OR 5, the function should return `n` as a string.
- Explicitly type your parameter and return type.

The following REPL examples demonstrate the expected functionality of your `summit` function:

```
1 >>> print(fizzbuzz(-2))
2 -2
3 >>> print(fizzbuzz(1))
4 1
5 >>> print(fizzbuzz(2))
6 2
7 >>> print(fizzbuzz(3))
8 fizz
```

```
1 >>> print(fizzbuzz(5))
2 buzz
3 >>> print(fizzbuzz(12))
4 fizz
5 >>> print(fizzbuzz(15))
6 fizzbuzz
7 >>> print(fizzbuzz(20))
8 buzz
```

6.1. Write your function definition here:

```
def fizzbuzz(n: int) -> str:
    if n % 3:
        if n % 5:
            return "fizzbuzz"
        else:
            return "fizz"
    elif n % 5:
        return "buzz"
    else:
        return str(n)
```

★ Note that there are many correct ways to write this function - this is just one example!

Question 7: CHALLENGE: Recursive Function Definition Writing Write a recursive function definition that returns the sum of all positive, even integers less than or equal to a given `int`. Your function definition should meet the following expectations:

- The function should be named `summit`, have one `int` parameter named `n`, and return an `int`.
- ✓ • If `n` is negative, the function should return `-1`.
- If `n` is positive, the function should return the sum of all positive, even integers less than or equal to `n`.
- Explicitly type your parameters and return types.
- Label your base case(s) and recursive case(s).

The following REPL examples demonstrate the expected functionality of your `summit` function:

```
1 >>> summit(-2)
2 -1
3 >>> summit(1)
4 0
5 >>> summit(2)
6 2
7 >>> summit(3)
8 2
```

```
1 >>> summit(4)
2 6
3 >>> summit(5)
4 6
5 >>> summit(6)
6 12
7 >>> summit(12)
8 42
```

7.1. Write your function definition here:

```
def summit(n: int) -> int:
    if n < 0: # Edge case -> we'll learn about this concept later!
        return -1
    elif n <= 1: # Base case
        return 0
    elif n % 2 == 1: # Recursive case
        return summit(n = n - 1)
    else: # Recursive case
        return n + summit(n = n - 2)
```

if n is odd, call summit again with an argument of n-1

if n is even and > 1, return the sum of n and the return value of calling summit with an argument of n-2 (the next lowest even int)

This page intentionally left blank. Do not remove from quiz packet.