**A black background with a black square

Description automatically generated with medium confidence**

**Assessment cover**

|  |  |  |  |
| --- | --- | --- | --- |
| Module No: | COMP4004 | Module title: | **Problem Solving and Programming** |

|  |  |
| --- | --- |
| Assessment title : | **Portfolio Exercises** |

|  |  |
| --- | --- |
| Due date and time**:** | **6th December 2024** |

|  |  |
| --- | --- |
| Estimated total time to be spent on assignment: | 60 hours per student |

**LEARNING OUTCOMES**

|  |
| --- |
| **On successful completion of this assignment, students will be able to achieve the module following learning outcomes (LOs):** *LO numbers and text copied and pasted from the module descriptor* |
| *LO 1: Design, implement and test computer programs, derived from application based case studies, implemented using an appropriate computer programming language* |
| *LO 2: Demonstrate a thorough understanding of the fundamental concepts of high level programming languages, including the syntax and semantics of constructs for input/output, control ﬂow and elementary data structuring.* |
|  |

|  |  |  |
| --- | --- | --- |
| **Engineering Council AHEP4 LOs assessed (from S2 2022-23)**  *LOs copied and pasted from the AHEP4 matrix (add rows as required)* | |  |
| **LO number** | **LO text** | **Met? (Y/N)** |
| **C3** | Select and apply appropriate computational and analytical techniques to model complex problems, recognising the limitations of the techniques employed |  |
| **C4** | Select and evaluate technical literature and other sources of information to address complex problems |  |
|  |  |  |

The use of AI tools is not allowed

**Statement of Compliance *(please tick to sign)***

I declare that the work submitted is my own and that the work I submit is fully in accordance with the University regulations regarding assessments *(*[*www.brookes.ac.uk/uniregulations/current*](http://www.brookes.ac.uk/uniregulations/current)*)*

COMP4004 Problem Solving & Programming Portfolio

# Week 2

## WK2EX9 [P] Postal Program (4 marks)

w = int(input("Enter weight (g): "))

h = int(input("Enter height (cm): "))

d = int(input("Enter diameter (cm): "))

if w<=2000 and h<=90 and (h+d\*2)<=104:

    print("Yes")

else:

    print("No")

## WK2EX10 [P] Overlapping Events (4 marks)

start\_a = int(input("Enter start hour for event A: "))

dur\_a = int(input("Enter duration of event A (in hours): "))

end\_a = start\_a+dur\_a

start\_b = int(input("Enter start hour for event B: "))

dur\_b = int(input("Enter duration of event B (in hours): "))

end\_b = start\_b+dur\_b

if start\_a < start\_b < end\_a or start\_a < end\_b < end\_a:

    print("Events overlap")

elif start\_b < start\_a < end\_b or start\_b < end\_a < end\_b:

    print("Events overlap")

else:

    print("Events do not overlap")

# Week 3

## WK3EX7 [P] Lojban Numbers (4 marks)

loj = {'0': 'no', '1': 'pa', '2': 're', '3': 'ci', '4': 'vo', '5': 'mu',

       '6': 'xa', '7': 'ze', '8': 'bi', '9': 'so'

       }

num = (input("Enter number: "))

num = num.lstrip('0')

if not num:

    print('no')

else:

    lojbnum = (loj[number] for number in num)

    print(''.join(lojbnum))

## WK3EX8 [P] Bingo Card (4 marks)

import random

i=0

b1 = []

b2 = []

b3 = []

b4 = []

b5 = []

bingo = range(11, 75)

while len(b1) != 5:

    r1 = random.choice(bingo)

    str1 = str(r1)

    if str1 in b1:

        continue

    else:

        b1.append(str1)

    i+=1

    if len(b1) == 5:

        print(' '.join(b1))

    while len(b2) != 5:

        r2 = random.choice(bingo)

        str2 = str(r2)

        if str2 in b1:

            continue

        elif str2 in b2:

            continue

        else:

            b2.append(str2)

            i+=1

            if len(b2) == 5:

                print(' '.join(b2))

        while len(b3) != 5:

            r3 = random.choice(bingo)

            str3 = str(r3)

            if str3 in b1:

                continue

            elif str3 in b2:

                continue

            elif str3 in b3:

                continue

            else:

                b3.append(str3)

                i+=1

                if len(b3) == 5:

                    print(' '.join(b3))

            while len(b4) != 5:

                r4 = random.choice(bingo)

                str4 = str(r4)

                if str4 in b1:

                    continue

                elif str4 in b2:

                    continue

                elif str4 in b3:

                    continue

                elif str4 in b4:

                    continue

                else:

                    b4.append(str4)

                    i+=1

                    if len(b4) == 5:

                        print(' '.join(b4))

                while len(b5) != 5:

                    r5 = random.choice(bingo)

                    str5 = str(r5)

                    if str5 in b1:

                        continue

                    elif str5 in b2:

                        continue

                    elif str5 in b3:

                        continue

                    elif str5 in b4:

                        continue

                    elif str5 in b5:

                        continue

                    else:

                        b5.append(str5)

                        i+=1

                        if len(b5) == 5:

                            print(' '.join(b5))

# Week 4

## WK4EX7 [P] count\_multiples (4 marks)

def count\_multiples(n,numlist):

    i = 0

    j = 0

    mult = len(numlist)

    while i < mult:

        a = numlist[i]

        if a%n == 0:

            j+=1

        i+=1

    return j

if \_\_name\_\_ == "\_\_main\_\_":

    test = count\_multiples(2, [25, 5, 1, 10, 3, 7])

    print(test)

## WK4EX8 [P] longest streak (4 marks)

def longest\_streak(word,word\_list):

    streak = 0

    max\_streak = 0

    for a in word\_list:

        if a == word:

            streak += 1

            max\_streak = max(max\_streak, streak)

        else:

            streak = 0

    return max\_streak

# Week 6

## WK6EX8 [P] words\_containing (5 marks)

def words\_containing(file,substring):

    contain = []

    sub = substring

    with open(file, 'r', encoding="utf-8") as file:

        for word in file:

            w = word.split()

            for words in w:

                if sub in words:

                    contain.append(words)

    return contain

## WK6EX9 [P] average\_rating (5 marks)

def average\_rating(file,candidate):

    with open(file, 'r', encoding="utf-8") as r\_file:

        r\_list = []

        zero = [0, 0]

        j = 0

        av = 0

        for line in r\_file:

            a = line.split()

            if candidate == a[1]:

                j+=1

                av+=int(a[2])

        if j == 0:

            return zero

        r\_list.append(j)

        r\_list.append(av/j)

        return r\_list

def print\_ratings(file):

    ratings = {}

    with open(file, 'r', encoding="utf-8") as r\_file:

        for line in r\_file:

            a = line.split()

            c = a[1]

            r = int(a[2])

            if c not in ratings:

                ratings[c] = {'total': 0, 'count': 0}

            ratings[c]['total'] += r

            ratings[c]['count'] += 1

    for c, d in ratings.items():

        av = d['total']/d['count']

        print(f"{c}: {av}")

print\_ratings('rating.txt')

# Week 7

## WK7EX10&11 [P] Marathon Program (8 marks)

def read\_file(filename):

    results = []

    with open(filename, 'r', encoding='utf-8') as file:

        for line in file:

            a = line.split(',')

            num = a[0]

            time = a[1]

            first = a[2]

            last = a[3]

            result = {'id': num, 'time': time, 'firstname': first, 'lastname': last}

            results.append(result)

        return results

def get\_secs(time):

    time\_split = time.split(':')

    seconds = int(time\_split[2]) + 60\*int(time\_split[1]) + 60\*60\*int(time\_split[0])

    return seconds

def get\_interval\_data(start\_secs, end\_secs, results):

    count = 0

    mean = 0

    for racer in results:

        time = racer['time']

        time\_secs = get\_secs(time)

        if start\_secs <= time\_secs <= end\_secs:

            count+=1

            mean+=time\_secs

        int\_mean = mean/count if count > 0 else 0

        interval = {'count': count, 'mean': int\_mean}

    return interval

# Week 8

## WK8EX9 [P] Inverting Triangle (4 marks)

from tkinter import \*

root = Tk()

canvas = Canvas(root,width=500,height = 500)

og\_points = [150, 0, 350, 0, 250, 500]

points = og\_points.copy()

def invert\_triangle(event):

    global points

    if points == og\_points:

        points[1] = og\_points[5]

        points[3] = og\_points[5]

        points[5] = og\_points[1]

    else:

        points = og\_points.copy()

    canvas.delete("all")

    canvas.create\_polygon(points, fill='magenta')

if \_\_name\_\_ == "\_\_main\_\_":

    canvas.pack(side = 'bottom')

    canvas.create\_polygon(points,fill = 'magenta')

    button = Button(root,text = 'Invert')

    button.pack(fill = X)

    button.bind('<Button-1>',invert\_triangle)

    root.resizable(width=False, height=False)

    root.mainloop()

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

## WK8EX10 [P] Press Release (4 marks)

from tkinter import \*

root = Tk()

canvas = Canvas(root,width=500,height = 500)

rectangle = [0, 0, 0, 0]

def press\_handler(event):

    xpress = event.x

    ypress = event.y

    rectangle[:] = [xpress, ypress, 0, 0]

def release\_handler(event):

    xrelease = event.x

    yrelease = event.y

    rectangle[2] = xrelease

    rectangle[3] = yrelease

    canvas.delete('all')

    canvas.create\_rectangle(rectangle, fill = 'lime')

if \_\_name\_\_ == "\_\_main\_\_":

    canvas.pack()

    canvas.bind('<Button-1>', press\_handler)

    canvas.bind('<ButtonRelease-1>', release\_handler)

    root.resizable(width=False, height=False)

    root.mainloop()

A screenshot of a computer

Description automatically generated