# Answer key

COM161 ( 61344 ) - Problem Solving for Computing. 2024-25.

# Week 16 Class Test 2 (60% of overall mark)

## Question 1

5 points

Modify the program you created for Practice Task 1 to find the sum of numbers which are a multiples of 5 up to a user-given value. Thus if the user entered 15 the program would return the number 30 (=5+10+15). You may assume that the user requests an upper number that is a multiple of 5.
What is the sum of the multiples of 5 from 5 to 215? ie 5+10+15+...215.

A 1180

B 2315

C 4730 Correct answer

D 6545

E 8265

## Question 2

5 points

This question is based on Practical Task 7. It is assumed that you have a program that will read books from a text file into a list of Book objects. The following line of code calculates the average price of the books once you have populated your list of Book objects.

```
print(f"The average price is: {Book.average_price(books):.2f}")
```

According write a function in your Book class that will calculate the average price of the books. Once you have done that, fill in the blanks. Please use the following text file *books1.txt* for the purposes of testing your code.

**books1.txt**

```
@classmethod
def average_price([Blank 1], books):
    total_price = sum(book.[Blank 2] for book in books)
    average_price = [Blank 3] / len(books)
    [Blank 4] average_price
```

The average price of the books is **[Blank 5]**

**Blank 1** cls Correct answer

Responses must match exactly and are case-sensitive

**Blank 2** price — Correct answer

Responses must match exactly and are case-sensitive

**Blank 3** total_price — Correct answer

Responses must match exactly and are case-sensitive

**Blank 4** return — Correct answer

Responses must match exactly and are case-sensitive

**Blank 5** 28.54 — Correct answer

Responses must match exactly

Scoring: Allow partial credit

## Question 3

**5 points**

Examine the function below, which searches for a specific number among a list of numbers. The function accepts two parameters; a list (some_list) followed by a target number (tar_number). The function processes these parameters and returns the appropriate Boolean value. Fill-in the missing elements, noting these are case sensitive (*worth 1/2 mark each*).

```
def search_list([Blank 1], [Blank 2]):
    for num in [Blank 3]:
        if num == [Blank 4]:
            return [Blank 5]
        else:
            return [Blank 6]
```

**Blank 1** some_list — Correct answer

Responses must contain the acceptable answers

**Blank 2** tar_number — Correct answer

Responses must contain the acceptable answers

**Blank 3** some_list — Correct answer

Responses must contain the acceptable answers

**Blank 4** tar_number — Correct answer

Responses must contain the acceptable answers

**Blank 5** True — Correct answer

Responses must contain the acceptable answers and are case-sensitive

**Blank 6** False — Correct answer

Responses must contain the acceptable answers and are case-sensitive

Scoring: Allow partial credit

## Question 4

**5 points**

Practice task 3 asked you to compute the value of n at which Algorithm A2 became more efficient than Algorithm A1,
The run time for algorithm A1 is found to have changed to

$T_1=0.008n^3$
while the function giving the time taken for algorithm A2 remains unchanged.
Modify your code to find the new value of n for which A2 becomes more efficient than A1
The value of n at which this first occurs is:

(A) 81

(B) 73

(C) 61

(D) 52

(E) 46 Correct answer

## Question 5

5 points

Examine the partial program below, which: (1) uses code to open a specified text file (available below), which contain a list of words, one per line and (2) displays a list of all the unique words found in the file.

**sample_words_b.txt**

Your task is to:
**(a)** identify the missing code
**(b)** extend the program to count the number of unique words found in the file and
**(c)** implement the letter_tally() function to count letter occurrences. Once you've done that, attempt to answer the questions below.

```
def uniqueWordFinder():
    dup_words = open('_____________.txt','r')
    set_List = set()
    for each_word in ___________:
        set_List.add(___________.rstrip('\n'))

    return set_List

# TODO1 – call the function & count the number of unique words found (don't forget to place the sample txt
file and your code within the same working folder).

# TODO2 – create a new function called letter_tally() that counts (tallies) the number of times a letter
appears in the Set, ignoring case
# sensitivity. The function should receive two parameters:
# --> 1. the Set returned from uniqueWordFinder() and
# --> 2. a letter of the alphabet e.g. "T".

# TODO3 – call letter_tally() and print the result
```

| Prompt | Answer |
|---|---|
| (1) How many unique words does the file contain? | 645 |

Partial: 50% Negative: 0%

| | |
|---|---|
| (2) How many times does the letter "t" appear in the Set of words? | 279 |

Partial: 50% Negative: 0%

**Distractors**

(1) 281

② 630

③ 584

④ 182

⑤ 183

⑥ 667

⑦ 732

⑧ 640

⑨ 301

⑩ 345

## Question 6

5 points

**Employee Work Hours Tracker**
This program tracks the total work hours of employees using a dictionary, where the keys represent employee names and the values store their accumulated hours. The program allows you to update an employee's hours, display all recorded hours, and check if any employee has exceeded a specific threshold of work hours.

**Select the missing code for each blank**. Worth 1/2 mark each.

```
def update_hours(hours, employee, time):
    """Updates work hours for an employee."""
    if employee in hours:
        hours[employee] += [Blank 1]
    else:
        hours[employee] = time

def display_hours(hours):
    """Displays the total work hours for all employees."""
    if hours:
        print("Work Hours:")
        for employee, time in [Blank 2]:
            print(f"{employee}: {time} hours")
    else:
        print([Blank 3])

# Create an empty work hours dictionary
hours = [Blank 4]

# Add or update hours for employees
update_hours(hours, 'Alice', 5)
update_hours(hours, 'Bob', 8)
update_hours(hours, 'Alice', 3)

# Display all work hours
[Blank 5]

# Check if an employee's hours exceed 10
if hours['Alice'] > [Blank 6]:
    print("Alice has exceeded 10 hours.")
```

**Prompt** **Answer**

| | |
|---|---|
| 1 Blank 1 | time |

**Partial: 16.66% Negative: 0%**

| | |
|---|---|
| 2 Blank 2 | hours.item() |

**Partial: 16.66% Negative: 0%**

| | |
|---|---|
| 3 Blank 3 | "No work hours recorded." |

**Partial: 16.66% Negative: 0%**

| | |
|---|---|
| 4 Blank 4 | {} |

**Partial: 16.66% Negative: 0%**

| | |
|---|---|
| 5 Blank 5 | display_hours(hours) |

**Partial: 16.66% Negative: 0%**

| | |
|---|---|
| 6 Blank 6 | 10 |

**Partial: 16.7% Negative: 0%**

**Distractors**

1. print_output()
2. get_info(hours)
3. range(len(list))
4. []

## Question 7

**5 points**

This question has five fill in the blanks.

Based on Practice Task 7, consider a class method for the Book class that will discount books by a particular author and show the overall discount for those books.

The call to the method could be as follows

```
Book.discount_books_by_author(books, "Casey Moore", 15)
```

Here all books in the books list written by Casey Moore will be discounted by 15% and would be given an output as follows.

*Discounted 'Book 224' by Casey Moore: £13.50 -> £11.47*
*Discounted 'Book 630' by Casey Moore: £9.98 -> £8.48*
*Discounted 'Book 908' by Casey Moore: £24.61 -> £20.92*
*Discounted 'Book 776' by Casey Moore: £27.27 -> £23.18*
*Discounted 'Book 646' by Casey Moore: £27.07 -> £23.01*
*Total discount applied to books by Casey Moore: £15.36*

Fill in the four blanks in the code below to achieve this

```
@classmethod
def discount_books_by_author(cls, books, author, discount_percentage):

    total_discount = [Blank 1]

    for book in books:
        if book.author == [Blank 2]:
            original_price = book.price
            discount_amount = book.price * (discount_percentage / 100)
            book.price -= [Blank 3]
            [Blank 4] += discount_amount
            print(f"Discounted '{book.title}' by {author}: £{original_price:.2f} -> £{book.price:.2f}")


    print(f"Total discount applied to books by {author}: £{total_discount:.2f}")
```

Using this text file *book1.txt* what is the total discount for books written by Sam Smith if a discount factor of 20% is applied

**books1.txt**

Total discount is **[Blank 5]**

**Blank 1** 0 Correct answer

Responses must match exactly

**Blank 2** author Correct answer

Responses must match exactly and are case-sensitive

**Blank 3** discount_amount Correct answer

Responses must match exactly and are case-sensitive

**Blank 4** total_discount Correct answer

Responses must match exactly and are case-sensitive

**Blank 5** 13.83 Correct answer

Responses must match exactly

Scoring: Allow partial credit

## Question 8

5 points

Study the example code presented below, which writes a dictionary object (called student_dict) to a binary before reading the binary file (called student_store.dat) back into a new dictionary object (called new_stud_dict). As with any pickling process, the structure of the binary data needs to be known by the programmer before it can be processed within the code; in this case to print the contents of the object.

Now, consider a different binary file which was used to encoded a 2d-list of bank accounts in the format:

```
bank_acc = [[5015687, 24400],[5014875, 13580]…]
```

where each sub list element stores the account number and account balance.
Your task is to write Python code to successfully read in the binary file (accessible here

**bank_account_d(2).dat**

) and implement appropriate functions to answer the following queries.

1. The number of bank accounts with an account balance of >= £40,000 is **[Blank 1]**
2. The account number of the bank account with the highest account balance is **[Blank 2]**
3. The combined total value those bank accounts with an account balance in excess of >= £60000 is **[Blank 3]**

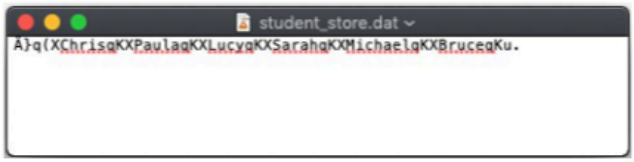
```
# dictionary of student and age
students_dict = {'Chris':21, 'Paula':22, 'Lucy':21, \
                 'Sarah':19, 'Michael':20, 'Bruce':21
                }

out_outfile = open('student_store.dat', 'wb')
pickle.dump(students_dict, out_outfile)
out_outfile.close()

input_file = open('student_store.dat', 'rb')
new_stud_dict = pickle.load(input_file)

for stud in new_stud_dict:
    print(stud, new_stud_dict[stud])
```

```
Chris 21
Paula 22
Lucy 21
Sarah 19
Michael 20
Bruce 21
```




**Blank 1** 606 Correct answer

Responses must contain the acceptable answers

**Blank 2** 1229307 Correct answer

Responses must contain the acceptable answers

**Blank 3** 32922799; 32,922,799; 32,922,799.00 Correct answer

Responses must contain the acceptable answers

Scoring: Allow partial credit

## Question 9

5 points

In Practice Task 7, you were tasked with creating an Ebook subclass, add another attribute called download_link and write an instance method to check to see if the download link contains "http" or not. Test the method to see if it works and then fill in the blanks for the instance method below.

```
[blank 1] check_download_link([blank 2]):

    [blank 3] [blank 4] in [blank 5]
```

| Prompt | Answer |
|---|---|
| 1 blank 1 | def |
| 2 blank 2 | self |

Partial: 20% Negative: 0%

Partial: 20% Negative: 0%

| | |
|---|---|
| ③ blank 3 | return |

Partial: 20% Negative: 0%

| | |
|---|---|
| ④ blank 4 | "http" |

Partial: 20% Negative: 0%

| | |
|---|---|
| ⑤ blank 5 | self.download_link |

Partial: 20% Negative: 0%

**Distractors**

① True

② False

③ cls

④ http

⑤ not

⑥ contains

⑦ method

## Question 10

5 points

The file

**optometry_q6(1).txt**

contains data to be read by your solution to Practice Task 2.
Modify your code to calculate the average age, correct to one decimal place, of the people with BLUE eyes

Ⓐ 35.4

Ⓑ 35.7

Ⓒ 35.0 Correct answer

Ⓓ 34.7

Ⓔ 35.8

## Question 11

5 points

The file

**optometry_q3(1).txt**

contains data to be read by your solution to Practice Task 2.
Run your code to find the percentage of people with GREEN eyes in the file.

A 52.1

B 20.9

C 32.7

D 9.1 Correct answer

E 33.3

## Question 12

5 points

What is the sum of the even numbers from 2 to 150? ie 2+4+6+8+...150?

A 3660

B 5700 Correct answer

C 8190

D 9120

E 3600

# Week 16 Class Test 2 (60% of overall mark)

## Question 1

5 points

Modify the program you created for Practice Task 1 to find the sum of numbers which are a multiples of 5 up to a user-given value. Thus if the user entered 15 the program would return the number 30 (=5+10+15). You may assume that the user requests an upper number that is a multiple of 5.
What is the sum of the multiples of 5 from 5 to 215? ie 5+10+15+...215.

A 1180

B 2315

C 4730

D 6545

E 8265

## Question 2

5 points

This question is based on Practical Task 7. It is assumed that you have a program that will read books from a text file into a list of Book objects. The following line of code calculates the average price of the books once you have populated your list of Book objects.

```
print(f"The average price is: {Book.average_price(books):.2f}")
```

According write a function in your Book class that will calculate the average price of the books. Once you have done that, fill in the blanks. Please use the following text file *books1.txt* for the purposes of testing your code.

**books1.txt**

```
@classmethod

def average_price([Blank 1], books):

    total_price = sum(book.[Blank 2] for book in books)

    average_price = [Blank 3] / len(books)

    [Blank 4] average_price
```

The average price of the books is **[Blank 5]**

**Blank 1**

**Blank 2**

**Blank 3**

**Blank 4**

**Blank 5**

## Question 3

5 points

Examine the function below, which searches for a specific number among a list of numbers. The function accepts two parameters; a list (some_list) followed by a target number (tar_number). The function processes these parameters and returns the appropriate Boolean value. Fill-in the missing elements, noting these are case sensitive (*worth 1/2 mark each*).

```
def search_list([Blank 1], [Blank 2]):
    for num in [Blank 3]:
        if num == [Blank 4]:
            return [Blank 5]
        else:
            return [Blank 6]
```

**Blank 1**

**Blank 2**

**Blank 3**

**Blank 4**

**Blank 5**

**Blank 6**

## Question 4

5 points

Practice task 3 asked you to compute the value of n at which Algorithm A2 became more efficient than Algorithm A1,
The run time for algorithm A1 is found to have changed to
$T_1=0.008n^3$
while the function giving the time taken for algorithm A2 remains unchanged.
Modify your code to find the new value of n for which A2 becomes more efficient than A1
The value of n at which this first occurs is:

A 81

B 73

C 61

D 52

E 46

## Question 5

5 points

Examine the partial program below, which: (1) uses code to open a specified text file (available below), which contain a list of words, one per line and (2) displays a list of all the unique words found in the file.

**sample_words_b.txt**

Your task is to:
**(a)** identify the missing code
**(b)** extend the program to count the number of unique words found in the file and
**(c)** implement the letter_tally() function to count letter occurrences. Once you've done that, attempt to answer the questions below.

```
def uniqueWordFinder():
    dup_words = open('_____________.txt','r')
    set_List = set()
    for each_word in ____________:
        set_List.add(____________.rstrip('\n'))

    return set_List

# TODO1 – call the function & count the number of unique words found (don't forget to place the sample txt
file and your code within the same working folder).

# TODO2 – create a new function called letter_tally() that counts (tallies) the number of times a letter
appears in the Set, ignoring case
# sensitivity. The function should receive two parameters:
# --> 1. the Set returned from uniqueWordFinder() and
# --> 2. a letter of the alphabet e.g. "T".

# TODO3 – call letter_tally() and print the result
```

| Prompt | Answer |
|---|---|
| ① How many unique words does the file contain? | 667 |
| ② How many times does the letter "t" appear in the Set of words? | 732 |
| | 630 |
| | 301 |
| | 279 |
| | 183 |
| | 345 |
| | 182 |
| | 640 |
| | 584 |
| | 281 |
| | 645 |

## Question 6

5 points

**Employee Work Hours Tracker**
This program tracks the total work hours of employees using a dictionary, where the keys represent employee names and the values store their accumulated hours. The program allows you to update an employee's hours, display all recorded hours, and check if any employee has exceeded a specific threshold of work hours.

**Select the missing code for each blank**. Worth 1/2 mark each.

```
def update_hours(hours, employee, time):
    """Updates work hours for an employee."""
    if employee in hours:
        hours[employee] += [Blank 1]
    else:
        hours[employee] = time

def display_hours(hours):
    """Displays the total work hours for all employees."""
    if hours:
        print("Work Hours:")
        for employee, time in [Blank 2]:
            print(f"{employee}: {time} hours")
    else:
        print([Blank 3])

# Create an empty work hours dictionary
hours = [Blank 4]

# Add or update hours for employees
update_hours(hours, 'Alice', 5)
update_hours(hours, 'Bob', 8)
update_hours(hours, 'Alice', 3)

# Display all work hours
[Blank 5]

# Check if an employee's hours exceed 10
if hours['Alice'] > [Blank 6]:
    print("Alice has exceeded 10 hours.")
```

| Prompt | Answer |
| --- | --- |
| ① Blank 1 | 10 |
| ② Blank 2 | [] |
| ③ Blank 3 | time |
| ④ Blank 4 | {} |
| ⑤ Blank 5 | hours.item() |
| ⑥ Blank 6 | display_hours(hours) |
| | "No work hours recorded." |
| | range(len(list)) |
| | print_output() |
| | get_info(hours) |

## Question 7

5 points

This question has five fill in the blanks.

Based on Practice Task 7, consider a class method for the Book class that will discount books by a particular author and show the overall discount for those books.

The call to the method could be as follows

```
Book.discount_books_by_author(books, "Casey Moore", 15)
```

Here all books in the books list written by Casey Moore will be discounted by 15% and would be given an output as follows.

*Discounted 'Book 224' by Casey Moore: £13.50 -> £11.47*
*Discounted 'Book 630' by Casey Moore: £9.98 -> £8.48*
*Discounted 'Book 908' by Casey Moore: £24.61 -> £20.92*
*Discounted 'Book 776' by Casey Moore: £27.27 -> £23.18*
*Discounted 'Book 646' by Casey Moore: £27.07 -> £23.01*
*Total discount applied to books by Casey Moore: £15.36*

Fill in the four blanks in the code below to achieve this

```
@classmethod
def discount_books_by_author(cls, books, author, discount_percentage):

    total_discount = [Blank 1]

    for book in books:
        if book.author == [Blank 2]:
            original_price = book.price
            discount_amount = book.price * (discount_percentage / 100)
            book.price -= [Blank 3]
            [Blank 4] += discount_amount
            print(f"Discounted '{book.title}' by {author}: £{original_price:.2f} -> £{book.price:.2f}")


    print(f"Total discount applied to books by {author}: £{total_discount:.2f}")
```

Using this text file *book1.txt* what is the total discount for books written by Sam Smith if a discount factor of 20% is applied

**books1.txt**

Total discount is **[Blank 5]**

**Blank 1**

**Blank 2**

**Blank 3**

**Blank 4**

**Blank 5**

## Question 8

**5 points**

Study the example code presented below, which writes a dictionary object (called student_dict) to a binary before reading the binary file (called student_store.dat) back into a new dictionary object (called new_stud_dict). As with any pickling process, the structure of the binary data needs to be known by the programmer before it can be processed within the code; in this case to print the contents of the object.

Now, consider a different binary file which was used to encoded a 2d-list of bank accounts in the format:

```
bank_acc = [[5015687, 24400],[5014875, 13580]…]
```

where each sub list element stores the account number and account balance.
Your task is to write Python code to successfully read in the binary file (accessible here

**bank_account_d(2).dat**

) and implement appropriate functions to answer the following queries.

1. The number of bank accounts with an account balance of >= £40,000 is **[Blank 1]**
2. The account number of the bank account with the highest account balance is **[Blank 2]**
3. The combined total value those bank accounts with an account balance in excess of >= £60000 is **[Blank 3]**

```
# dictionary of student and age
students_dict = {'Chris':21, 'Paula':22, 'Lucy':21, \
                 'Sarah':19, 'Michael':20, 'Bruce':21
                }

out_outfile = open('student_store.dat', 'wb')
pickle.dump(students_dict, out_outfile)
out_outfile.close()

input_file = open('student_store.dat', 'rb')
new_stud_dict = pickle.load(input_file)

for stud in new_stud_dict:
    print(stud, new_stud_dict[stud])
```

```
Chris 21
Paula 22
Lucy 21
Sarah 19
Michael 20
Bruce 21
```

student_store.dat

```
A}q(XChrisqKXPaulaqKXLucyqKXSarahqKXMichaelqKXBruceqKu.
```

**Blank 1**

**Blank 2**

**Blank 3**

## Question 9

5 points

In Practice Task 7, you were tasked with creating an Ebook subclass, add another attribute called download_link and write an instance method to check to see if the download link contains "http" or not. Test the method to see if it works and then fill in the blanks for the instance method below.

```
[blank 1] check_download_link([blank 2]):

    [blank 3] [blank 4] in [blank 5]
```

| Prompt | Answer |
| --- | --- |
| 1 blank 1 | self |
| 2 blank 2 | http |

| | |
|---|---|
| ③ blank 3 | contains |
| ④ blank 4 | cls |
| ⑤ blank 5 | "http" |
| | def |
| | False |
| | method |
| | True |
| | return |
| | self.download_link |
| | not |

## Question 10

5 points

The file

**optometry_q6(1).txt**

contains data to be read by your solution to Practice Task 2.
Modify your code to calculate the average age, correct to one decimal place, of the people with BLUE eyes

Ⓐ 35.4

Ⓑ 35.7

Ⓒ 35.0

Ⓓ 34.7

Ⓔ 35.8

## Question 11

5 points

The file

**optometry_q3(1).txt**

contains data to be read by your solution to Practice Task 2.
Run your code to find the percentage of people with GREEN eyes in the file.

Ⓐ 52.1

(B) 20.9

(C) 32.7

(D) 9.1

(E) 33.3

## Question 12

5 points

What is the sum of the even numbers from 2 to 150? ie 2+4+6+8+...150?

(A) 3660

(B) 5700

(C) 8190

(D) 9120

(E) 3600