UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

FINAL EXAMINATION, DECEMBER 2024

DURATION: 2½ hours

ESC 180 H1F — Introduction to Computer Programming

Calculator Type: None
Exam Type: B

Aids allowed: reference sheet distributed with the exam
Examiner(s): M. Guerzhoy

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|

UTORid: _____

UofT email: _____@mail.utoronto.ca

Family Name(s): _____

Given Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This final examination paper consists of 8 questions on 15 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy is complete, and fill in the identification section above.*

Answer each question directly on this paper, in the space provided. Use the pages at the end of the exam for extra space. If you use extra pages, indicate that you have done so in the space under the question.

Write up your solutions carefully! Comments and docstrings are *not* required to receive full marks, except where explicitly indicated otherwise. However, they may help us mark your answers, and part marks *might* be given for partial solutions with comments clearly indicating what the missing parts should accomplish.

When you are asked to write code, *no* error checking is required: you may assume that all user input and argument values are valid, except where explicitly indicated otherwise.

Use the Python 3 programming language. You may not `import` any module except `math`, unless otherwise specified.

You **must** write your student number on every odd-numbered page of the exam (except empty pages). **Failure to so will result in a 3-point penalty.**

MARKING GUIDE

Q1: _____/ 10
Q2: _____/ 15
Q3: _____/ 10
Q4: _____/ 10
Q5: _____/ 15
Q6: _____/ 20
Q7: _____/ 10
Q8: _____/ 10

TOTAL: _____/100

Good Luck!

Question 1. [10 MARKS]

In a basket of socks under the Christmas tree, each sock has a matching pair of the same colour, except for one sock whose match is missing. Given a non-empty list of strings, `socks`, where each string represents the colour of a sock, write a function to identify the colour that only appears once in the list. For example, `find_lonely_sock(["red","blue","blue","green","red"])` should return `"green"`.

```
def find_lonely_sock(socks):
```

Question 2. [15 MARKS]

In Santa's workshop, elves are working on toys. A dictionary `tasks` contains the toys as keys and the names of the elves working on them as values. For example, the dictionary `tasks` might look like this:

```
tasks = {
    "dolls": "Dewdrop",
    "trains": "Frostleaf",
    "robots": "Dewdrop",
    "puzzles": "Comet",
    "compilers": "Mike",
    "bridge kits": "Evan"
}
```

Part (a) [5 MARKS]

Write a function that takes in a dictionary like `tasks` and the name of an elf, and returns the number of toys that the elf is working on. For example, `num_tasks(tasks, "Dewdrop")` should return 2.

```
def num_tasks(tasks, elf_name):
```

Part (b) [10 MARKS] Write a function that takes in a dictionary like `tasks` and returns the name of the elf working on the most toys. Assume there is only one elf who is working on the most toys. For example, `most_tasks(tasks)` should return `"Dewdrop"`.

```
def hardest_worker(tasks):
```

Question 3. [10 MARKS]

A dictionary contains students' marks, with keys representing student numbers and values representing lists of percentage grades. For example:

```
student_grades = {"1005235672": [75, 66, 91], "1006249817": [99, 98, 95, 100], ...}
```

The GPA is computed by converting each percentage grade to a grade point and then averaging the grade points. The grade point conversion, for the purposes of this question, is as follows:

Percentage Grade	Grade Point
85-100	4.0
80-84	3.7
70-79	3.0
50-69	1.9
0-49	0.0

Write a function that takes a dictionary in the format described above, and returns the average GPA across all students. Hint: you may like to compute the GPA for each student before averaging the GPAs.

```
def get_avg_gpa(student_grades):
```

Question 4. [10 MARKS]

Recall that in lecture, we estimated the value of π by computing the area of the unit quartle-circle. We did that by generating random points in the unit square and counting the number of points that fall within the unit quarter-circle. We then used the ratio of the number of points in the quarter-circle to the total number of points to estimate the area of the quarter-circle, and hence the value of π.

Write code that estimates how many random points are needed in order to estimate π to within an error of 0.01 (i.e., the error should be smaller than that). You must do this by actually generating enough points and computing π using this method. Your code should print the estimated number of points needed.

You may `import random`, `math` (as for any question), and `numpy`. You are allowed to only simulate one experiment in order to get a reasonable estimate. Any reasonable solution will be accepted.

Question 5. [15 MARKS]

The current set up booked seats to see the *Nutcracker* is described by a *list of lists*, where each list represents a row. A booked seat is represented by `1` and an empty seat is represented by `0`. For example,

```
seats = [[1,0,1,1,1],
         [1,0,0,0,0],
         [0,0,1,0,0]]
```

shows a theatre with 3 rows and five seats per row. A booking agency wishes to determine the maximum party size that can be seated adjacent to each other in the same row. Write a function to return this maximum number. For the above example, the maximum party size is 4, because there are four empty seats on the very right of the second row. Note that you also need to consider situations where the largest party can be seated in the middle of a row.

```
def maximum_party_size(seats):
```

Question 6. [20 MARKS]

A palindrome is a word that reads the same forward and backward. Some examples are: `"racecar"`, `"level"`, `"kayak"`, and `"refer"`.

In The Palindrome Game, two players take turns removing letters from the start and the end of a string. The player who makes the string a palindrome first wins. An example of game-play between a computer player and a human player is shown below:

```
Starting string: cracecarab
Computer player action: left
Current string: racecarab
Human player action: right
Current string: racecara
Computer player action: right
Current string: racecar
Computer player wins!
```

Note: the sub-questions of this question are independent. You can receive full marks for a given sub-question even if your other sub-questions are incorrect.

Part (a) [2 MARKS] Write a function that returns `True` if a string is a palindrome and false otherwise. Note that a single letter (e.g., "a") is considered a palindrome. This function does not need to be recursive.

```
def is_palindrome(game_str):
```

Part (b) [12 MARKS] Write a recursive function in Python that takes the current game string as input and determines whether a player is guaranteed to win with optimal play if it is their move and they start with the string `game_str`. The function should return `True` if the player is guaranteed to win and `False` otherwise. You may assume that the input string is non-empty and contains only lowercase letters.

```
def can_win(game_str):
```

Part (c) [6 MARKS] Write code that allows a human user to play against your optimal computer player in The Palindrome Game. Get the starting string from the user, ask the user who starts, and then have the computer and human players take turns removing letters from the start and end of the string.

You may call your `is_palindrome(game_str)` and/or `can_win(game_str)` functions here and assume both function correctly. You should use `input(prompt)` to get input from the human player.

```
if __name__ == "__main__":
```

Question 7. [10 MARKS]

The left-hand column in the table below contains different pieces of code that work with `float n`. In the right-hand column, give the asymptotic tight upper bound on the worst-case runtime complexity of each piece of code, using Big O notation. Assume that arithmetic operations such as `+`, `*`, and `//` take constant time. Give a very brief justification for your answer. Answers with no or obviously incorrect justification will receive no marks, but the only requirement for full marks is that we can tell you understand the idea.

Code	Complexity

`def f(n):`
`  s = 1`
`  for i in range(int(n)):`
`  for j in range(int(n**0.5)):`
`  s += 1`
`f(n)`	

`def g1(n):`
` if n == 0:`
`  return 1`
` return g1(n//2) + g1(n//2)`
 	
`g1(n)`	

`def g2(n):`
`  if n == 0:`
`  return 1`
`  return 4*g2(n//2)`
 	
`g2(n)`	

`def h(n, memo={}):`
`  if n <= 1:`
`  return n`
`  if n not in memo:`
`  memo[n] = h(n - 1, memo) + h(n - 2, memo)`
`  return memo[n]`
 	
`h(n)`	

`def a(n):`
`  i = 1`
`  while i < n:`
`  i *= 2`
 	
`a(n)`	

Question 8. [10 MARKS]

Each of the subquestions in this question contains a piece of code. Treat each piece of code independently (i.e., code in one question is not related to code in another), and write the expected output for each piece of code. If the code produces an error, write down the output that the code prints before the error is encountered, and then write "ERROR." You do not have to specify what kind of error it is. Assume Python version 3.7+. No justification is required.

Code	Output

`L = [0, 1]`
`def g(M):`
`  global L`
`  L.append(2)`
`  return M`
`def f(L):`
`  L = L[:]`
`  L.append(3)`
`  g(L)`
`  return L`
`M = f(L)`
`print(L)`
`print(M)`	

`s = "abcd"`
`s[2] = "e"`
`s[3] = "f"`
`print(s)`	

`L = [0, [1, 2]]`
`M = L`
`M[1][0] = L[1][1]`
`L[0] = 3`
`L.append(4)`
`print(L)`
`print(M)`	

`def f(d):`
`  d[1] = "a"`
`  d1[2] = "b"`
`  d2[3] = d`
`d1 = {}`
`d2 = {4: d1}`
`f(d1)`
`print(d1)`
`print(d)`	

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.