Lab Test 1 [25 marks]

Coding time: 1 hour and 30 mins

General Instructions:

1. You will take the lab test on your personal laptop.

2. You are not allowed to communicate with anyone or access any websites during the test.

3. You may refer to any file on your laptop during the test.

4. Make sure your code can generate exactly the same output as we show in the sample runs. You may be penalized for missing spaces, missing punctuation marks, misspellings, etc. in the output.

5. Do not hardcode. We will use different test cases to test and grade your solutions.

6. Follow CS101 code conventions (e.g. naming functions and variables) or risk a 3-mark penalty on your final score.

7. C source file that cannot be compiled will NOT be marked and hence you will be awarded 0 marks. You may wish to comment out the parts in your code which cause compile errors.

8. Include your name as author in the comments of all your submitted source files. For example, include the following block of comments at the beginning of each source file you need to submit.

```
/*
 * Name : SEE Hao Wan
 * Email: haowan.see.2022
 */
```

Instructions on how to submit your solutions:

1. When the test ends, zip up all the files required for submission in a zip archive. The name of the zip archive should be your email ID. For example, if your email is haowan.see.2022@scis.smu.edu.sg, you should name the archive as haowan.see.2022. You may be penalized for not following our instructions.

2. Submit your solutions as a single zip file to eLearn Assignments.

Question 1 (Difficulty Level: *) [8 marks]

Implement a function called get_intensity. It takes in 3 parameters:

- `training_heart_rate` (type: `int`)
- `resting_heart_rate` (type: `int`)
- `age` (type: `int`)

This function returns the intensity of the activity (e.g. eating, walking, running, tennis) . The intensity of an activity is classified as follows:

- Sedentary(S): below 57%
- light (L): between 57% (inclusive) to 64% (exclusive)
- moderate(M): between 64% (inclusive) to 75% (exclusive)
- vigorous(V): between 75% (inclusive) to 95% (exclusive)
- high (H): 95% and above.

The intensity of an activity is calculated as follows:

$$intensity\ (\%) = \frac{training\ heart\ rate - resting\ heart\ rate}{220 - age - resting\ heart\ rate} \times 100$$

Example 1

If the function is invoked like this:

```
printf("%c\n", get_intensity(100, 65, 40));
```

the statement generates the following output:

```
S
```

Note:

1. The intensity is (100 - 65)/(220 - 40 - 65) x 100 = 30.4%

Example 2

If the function is invoked like this:

```
printf("%c\n", get_intensity(124, 54, 45));
```

the statement generates the following output:

```
L
```

Note:

2. The intensity is (124 - 54)/(220 - 45 - 54) x 100 = 57.9%

Question 2 (Difficulty Level: **) [5 marks]

The Collatz conjecture is a conjecture in mathematics which states that no matter what value of n, if the below sequence is followed then the sequence will always reach 1.

if n is even, $f(n) = \frac{n}{2}$

if n is odd, $f(n) = 3n + 1$

For example, if n is 10, then the sequence is as follows:

Step 1(n is even): **10** / 2 = 5
Step 2(n is odd): 3 x **5** + 1 = 16
Step 3(n is even): **16** / 2 = 8
Step 4(n is even): **8** / 2 = 4
Step 5(n is even): **4** / 2 = 1

It takes 3 steps to reach the value of 8, and 5 steps to reach the value of 1.

Implement a function called `get_num_steps`. It takes in two parameters:

- `n` (type: `int`): You may assume that n is a positive integer.
- `look_for` (type: `int`): the number we are looking for.

This function returns the number of steps that it takes to reach the first occurrence of `look_for`. If look_for is not in the sequence (i.e. it did not find `look_for` before reaching 1), this function returns -1. **It should work for all possible input values of n.**

Example 1

If the function is invoked like this:

```
printf("%d\n", get_num_steps(10, 16));
```

the statement generates the following output:

```
2
```

Note:

1. The sequence is 10, 5, 16, 8, 4, 2, 1.
2. It takes 2 steps to reach 16.

Example 2

If the function is invoked like this:

```
printf("%d\n", get_num_steps(3, 1));
```

the statement generates the following output:

```
7
```

Note:

1. The sequence is 3, 10, 5, 16, 8, 4, 2, 1.
2. It takes 7 steps to reach 1.

Example 3

If the function is invoked like this:

```
printf("%d\n", get_num_steps(8, 3));
```

the statement generates the following output:

```
-1
```

Note:

1. The sequence is as follows: 8, 4, 2, 1
2. The number 3 is not located in the sequence.

Example 4

If the function is invoked like this:

```
printf("%d\n", get_num_steps(123, 123));
```

the statement generates the following output:

```
0
```

Note:

1. 3 (`look_for`) is the `n` value.

Question 3 (Difficulty Level: **) [4 marks]

A new digital bank recently launched a new credit card, Faith, in collaboration with DecentlyPriced Group (DPG). A user can earn cashback by paying with the Faith credit card

- at DPG merchants, or
- at non-DPG merchants. If the user meets the monthly or quarterly minimum eligible amount for spendings at non-DPG merchants, the user will get additional bonus rates for DPG payments.

The cashback rates are as follow:

1. The user gets a 1% cashback for all spendings at non-DPG merchants.
2. For spending at DPG merchants, the rate is as follow:

Component	DPG cashback (%)	Minimum Spending	Cap (on cashback) per month
base	2	Not Applicable.	$100
month	7.5	$200 at non-DPG merchants per month	$150
quarter	9	$200 at non-DPG merchants per month for 3 consecutive months	$250

For example, if a user spends $200 at non-DPG merchants, and $2,050 at DPG merchants each month for 3 consecutive months, then the monthly cashback will be:

Month	Cashback
1	**base:** 0.02 x $2,050 (DPG) + 0.01 x $200(non-DPG) = $43 **month:** 0.075 x $2,050 = $153.75 **Total:** $43 (base) + $153.75 (month, capped at $150) = **$193**
2	**base:** 0.02 x $2,050 (DPG) + 0.01 x $200(non-DPG) = $43 **month:** 0.075 x $2,050 = $153.75 **Total:** $43 (base) + $153.75 (month, capped at $150) = **$193**
3	**base:** 0.02 x $2,050 (DPG) + 0.01 x $200(non-DPG) = $43 **month:** 0.075 x $2,050 = $153.75 **quarter** 0.09 x $2,050 = $184.50 **Total:** $43(base) + $153.75 (month, capped at $150) + $184.50 (quarter) = **$377.50**

Implement a function called `get_cashback`. It takes in two parameters:

- `spendings` (type: `double[]`): It records the spendings at DPG and non-DPG merchants each month in the following order

```
{ dpg_spending_1, non_dpg_spending_1,
  dpg_spending_2, non_dpg_spending_2,
  ..,
  dpg_spending_n, non_dpg_spending_n}
```

  where
  - `dpg_spending_n` represents the spending at DPG merchants in the n-th month, and
  - `non_dpg_spending_n` represents the spending at non-DPG merchants in the n-th month.
- `num` (type: `int`): It represents the number of months recorded for the user. If the value is 2, then there are 4 values in `spendings`.

This function returns the total amount of cashback **in cents**. The value is rounded to the nearest cent.

Note:

1. To round a number to the nearest whole number, you have to look at the first digit after the decimal point.
   a. If this digit is less than 5 (1, 2, 3, 4) we don't have to do anything. For example, 3.1 is rounded to 3.
   b. If the digit is 5 or greater (5, 6, 7, 8, 9) we must round up. For example, 3.6 is rounded to 4.

Example 1: base cashback

If the function is invoked like this:

```
double spendings[] = {100, 199};
printf("%d\n", get_cashback(spendings, 1));
```

the statement generates the following output:

```
399
```

Note:

1. The user did not meet the monthly eligible spending (i.e. no additional bonus) and this is his/her first month spending (i.e. no quarter bonus).
2. The cashback is 0.02 x $100 + 0.01 x $199 = $3.99 (399 cents).

Example 2: base with monthly cashback

If the function is invoked like this:

```
double spendings[] = {100, 300};
printf("%d\n", get_cashback(spendings, 1));
```

the statement generates the following output:

```
1250
```

Note:

3. The user met the monthly eligible spending (additional bonus). There is no quarter bonus.
4. The cashback is $5 + $7.50 = $12.50 (i.e. 1250 cents)
   a. base rate: 0.02 x $100 + 0.01 x $300 = $5
   b. bonus rate: 0.075 x $100 = $7.50

Example 3: base with month and quarter cashback

If the function is invoked like this:

```
double spendings[] = {100, 300, 2560, 345, 3000, 200, 100, 200};
printf("%d\n", get_cashback(spendings, 4));
```

the statement generates the following output:

```
69965
```

Note:

1. The cashback is (12.50 + 204.65 + 462 + 20.5) = $699.65 (69965 cents)

mth	Spending		Cashback	
1	DPG: non-DPG:	100 300 **(>= 200)**	base: month bonus: quarter bonus: Total:	0.02 x $100 + 0.01 x $300 = $5.00 0.075 x $100 = $7.50 N/A $5.00 + $7.50 = **$12.50**
2	DPG: non-DPG:	2560 345 **(>= 200)**	base: month bonus: quarter bonus: Total:	0.02 x $2560 + 0.01 x $345 = $54.65 0.075 x $2560 = $192 (exceeded cap of $150) N/A $54.65 + $150 = **$204.65**
3	DPG: non-DPG:	3000 200 **(>= 200)**	base: month bonus: quarter bonus: Total:	0.02 x $3000 + 0.01 x $200 = $62 0.075 x $3000 = $225 (exceeded cap of $150) 0.09 x $3000 = $270 (exceeded cap of $250) $62 + $150 + $250 = **$462**
4	DPG: non-DPG:	100 200 **(>= 200)**	base: month bonus: quarter bonus: Total:	0.02 x $100 + 0.01 x $200 = $4 0.075 x $100 = $7.50 0.09 x $100 = $9 $4 + $7.50 + $9 = **$20.50**

Question 4 (Difficulty Level: **) [4 marks]

The University Admission Score (UAS) is the sum of all the rank points (RP) for all subjects taken at 'A' levels. Each subject is either taken at H1 or H2 level. Implement a function called **`calculate_uas`**. It takes parameters that represent the grades taken by a student at 'A' level in the order as listed below:

- `h2_mods`(type: `char[]`): the student grades for his/her H2 subjects. It will be an array of length 4. If there is a `content` subject, it will be of length 3.
- `content` (type: `char`): the student grade for the H1 content subject. If the value is `'-'`, the student did another H2 subject in place of his/her h1 content subject. (see point 1 below).
- `gp` (type: `char`): the grade for H1 General Paper (GP)
- `pw` (type: `char`): the grade for H1 Project Work (PW)
- `mtl` (type: `char`): the grade for H1 Mother Tongue Language (MTL). For example, Chinese and Malay. If the value is `'-'`, it means that the student did not take this subject.

The below table shows the rank points for each grade. For example,

- getting an A for a H2 subject will give the student 20 rank points.
- getting an A for a H1 subject will give the student 10 rank points.

Grade	Rank Points(H2 Subject)	H1 Rank Points (H1 subject)
A	20	10
B	17.5	8.75
C	15	7.5
D	12.5	6.25
E	10	5
S	5	2.5
U	0	0

1. The formula is either:
   - (3 `h2_mods` + `content` + `gp` + `pw`), or
   - (3 `h2_mods` + `content` + `gp` + `pw` + `mtl`)/100 x 90

   For students with MTL, the higher of the 2 UAS is used.

   For students with 4 H2, the student's weakest H2 subject will be computed as a H1 content subject. This means the student's rank points achieved for that H2 subject will be halved (calculated as /10 instead of /20). For example, if the user scored A (20), C (15), C (15), E (10). The E grade will be treated as a H1 content subject with 5 rank points.

2. The maximum UAS of a student is 90 points.

Example 1

If the function is invoked like this:

```
char h2_mods[] = {'A', 'A', 'A', 'B'};
char content = '-'; // took 4 H2 subjects
char gp = 'A';
char pw = 'A';
char mtl = 'A';
printf("%.2lf\n", calculate_uas(h2_mods, content, gp, pw, mtl));
```

the statement generates the following output:

```
88.88
```

Note:

1. The student took 4 H2 modules without h1 (content is `'-'`).
2. The 3 best H2 subjects gave the student 60 points (20 + 20 + 20).
3. The weakest H2 subject is treated as the content subject at H1 level (8.75).
4. The UAS will be the higher of the two (with and without MTL), i.e. 88.875.
    a. Total (with MTL):
    $\frac{100}{90} \times [20 + 20 + 20 + 8.75(content) + 10(gp) + 10(pw) + 10(mtl)] = 88.875$
    b. Total (without MTL):
    $20 + 20 + 20 + 8.75(content) + 10(gp) + 10(pw) = 88.75$

Example 2

If the function is invoked like this:

```
char h2_mods[] = {'A', 'A', 'A'};
char content = 'B';
char gp = 'A';
char pw = 'B';
char mtl = '-';
printf("%.2lf\n", calculate_uas(h2_mods, content, gp, pw, mtl));
```

the statement generates the following output:

```
87.50
```

Note:

1. The student took 3 H2 modules, and did not take MTL (i.e. '–')
2. H2 content subjects scored 60 points (20 + 20 + 20).
3. The UAS is 20 + 20 + 20 + 8.75(content) + 10 (gp) + 8.75(pw) = 87.5

Example 3

If the function is invoked like this:

```
char h2_mods[] = {'A', 'C', 'B'};
char content = 'B';
char gp = 'A';
char pw = 'B';
char mtl = 'D';
printf("%.2lf\n", calculate_uas(h2_mods, content, gp, pw, mtl));
```

the statement generates the following output:

```
80.00
```

Note:

1. The student took 3 H2 modules.
2. H2 content subjects scored 52.5 points (20 + 17.5 + 15).
3. The UAS will be the higher of the two, i.e. 80
    a. with MTL:
    $\frac{100}{90} \times [20 + 15 + 17.5 + 8.75\,(content) + 10\,(gp) + 8.75\,(pw) + 6.25\,(mtl)] = 77.625$
    b. without MTL:
    $20 + 15 + 17.5 + 8.75(content) + 10\,(gp) + 8.75\,(pw) = 80$

Question 5 (Difficulty Level: ***) [4 marks]

Implement the function called `align`. It takes in two parameters:

- `num1` (type: `long`): Assume positive integer, i.e. greater than 0.
- `num2` (type: `long`): Assume positive integer, i.e. greater than 0.

It returns a new number (type: `long`) formed by first aligning the leftmost digits in `num2` that are found in `num1` in the same position. This alignment stops when we reach a digit in `num2` (starting from the left and moving right) that cannot be found in `num1` with the same digits to its left in both `num1` and `num2`.

After aligning, a minimum number of zeros are used to fill up all digits in the new number to maintain the alignment with `num1`.

See the examples below.

Example 1

If the function is invoked like this:

```
long num1 = 512345;
long num2 = 25191879;
printf("%d\n", align(num1, num2));
```

the statement generates the following output:

```
2005
```

Note:

1. We managed to align the digits 2 and 5 in num2 to num1 as follows:

   ```
   num1:   51234 5
   num2:     2   5191879
   result:   2005
   ```

   The alignment stops at 5 because the digit 1 is not located to the right of the digit 5 in num1.

2. The number 2005 is returned by inserting two zeros to replace the two additional digits (34) in num1.

Example 2

If the function is invoked like this:

```
long num1 = 934678;
long num2 = 2934;
printf("%d\n", align(num1, num2));
```

the statement generates the following output:

```
0
```

Note:

1. There is no alignment since the digit 2 in num2 is not found in num1.
2. The value 0 is thus returned.

Example 3

If the function is invoked like this:

```
long num1 = 934678;
long num2 = 769438;
printf("%d\n", align(num1, num2));
```

the statement generates the following output:

```
70
```

Note:

1. We managed to align the digit 7 in num2 to num1 as follows:

```
num1:          93467 8
num2:              7 69438
result:            70
```

   The digit 8 is not considered since the digit 6 cannot be located to the right of the digit 7 in num1.

Example 4

If the function is invoked like this:

```
long num1 = 934078;
long num2 = 940923;
printf("%d\n", align(num1, num2));
```

the statement generates the following output:

```
904000
```

Note:

1. We managed to align the digits 9, 4 and 0 in num2 to num1 as follows:

```
num1:       934078
num2:       9 40  923
result:     904000
```

Example 5

If the function is invoked like this:

```
long num1 = 123456;
long num2 = 24;
printf("%d\n", align(num1, num2));
```

the statement generates the following output:

```
20400
```

Note:

2. We managed to align all digits in num2 to num1 as follows:

```
num1:       123456
num2:        2 4
result:      20400
```

END

EMPTY PAGE
USE THIS PAGE AS SCRATCH PAD

EMPTY PAGE
USE THIS PAGE AS SCRATCH PAD

EMPTY PAGE
USE THIS PAGE AS SCRATCH PAD