CITY UNIVERSITY OF HONG KONG

Course code & title	:	CS2310 Computer Programming
Session	:	Semester B 2020/21
Time allowed	:	Two Hours

This paper has 7 pages (including this cover page).

---

1. This paper has **4** questions
2. Do not include another additional library
3. Download "Final.cpp" from Canvas, and re-name it using your student ID as "YourStudentID.cpp"
4. Write all your solutions in "YourStudentID.cpp"
5. Submit "YourStudentID.cpp" (.cpp file only) to Canvas

---

*This is an **open-book** examination.*

Candidates are allowed to use the following materials/aids:

Materials posted on Canvas for this course

Materials/aids other than those stated above are not permitted. Candidates will be subject to disciplinary action if any unauthorized materials or aids are found on them.

Q1: [Programming Style and Basic Programming] (25%)

Write a program to accept a **positive** integer N ($N \geq 3$) and print a **horizontally flipped** letter Z as shown in the following examples, such that both the *length* and the *height* of the printed shape are N. The *length* and *height* are measured by the **number** of stars ("*").

- There is **NO space** between two consecutive star ("*") positions **in each row**.
  - In the **second** row of Example-1, the printed shape contains only one star ("*") and there is one space in front of this star.
- There is **NO empty line** between any **two consecutive rows**.

- We assume N is already an integer, but you need to check whether $N \geq 3$.

Note: In the following examples, there is some distance between two consecutive rows. This is the line spacing distance in the text editor that cannot be avoided. When you write the codes, please **do NOT add** a newline between two rows.

Example-1 (Input is underlined):

```
Please input the integer N: 4
****
 *
  *
****
```

Example-2 (Input is underlined):

```
Please input the integer N: 5
*****
 *
  *
   *
*****
```

Example-3 (Inputs are underlined):

```
Please input the integer N: -1
Please input the integer N: 2
Please input the integer N: 3
***
 *
***
```

Q2: [Program Development] (25%)

A **recursive function** is defined as follows: $f(0) = 1$, $f(1) = 2$, and

- $f(n) = 2 \times f(n-1) + f(n-2),\ if\ n \geq 2.$

Part (a): Write a program to conduct the following operations:

- Read a non-negative (*data type:* ***int***) integer n ($n \geq 2$) from an **input file** "input.txt" (this file contains one non-negative integer number only) and show it on the screen.
- Write a **recursive function** to calculate the result of $f(n)$. Function's return type is **int**.
- Output the result of $f(n)$ on the screen.

Part (b): For the result of $f(n)$, you need to further get each of its digits **sequentially** and store them in an integer (**int-type**) array using **dynamic memory allocation**. The array size should be the same as the number of digits in the result of $f(n)$. For example, if the result of $f(n)$ is 1277686868, the size of the array is ten and the digits should be stored as follows:

Digit:	1	**2**	**7**	7	**6**	8	6	8	6	8
Digit's Index:	0	**1**	**2**	3	**4**	5	6	7	8	9

Next, your program further asks the user to input two integer values, stored in `start_idx` and `end_idx` two variables (*data type:* ***int***), respectively. We can then obtain a number, stored in `num` (*data type:* ***int***), starting from `start_idx` and ending at `end_idx` in the dynamic array, e.g.,

- If `start_idx` is 1 and `end_idx` is 2 in the table above, `num` is 27.
- If `start_idx` is 4 and `end_idx` is 4 in the table above, `num` is 6.

- Assume 0 ≤ start_idx ≤ end_idx < Digit # in $f(n)$. **No need** to check their correctness.
- We assume that `start_idx` and `end_idx` are properly selected, so that num > 0.

Print the value of this selected number `num` on the screen.

Part (c): Your program determines whether `num` has a special property, i.e., `num` is equal to the **sum** of all its positive **factors** excluding `num` itself. For example, $28 = 1 + 2 + 4 + 7 + 14$.

- If `num` has this property, print "This number is a special number", and also print how many of these factors are **prime numbers**, e.g., for 28, factors 2 and 7 are prime numbers.
- If `num` does not have this property, simply print "This number is not a special number".

Note that:

- If `a` is one factor of `num` in this question, num % a is 0 and 1 ≤ a < num.
- If `a` is one prime number, 2 ≤ a and `a` has **NO** other positive factors between 1 and itself.

Implement the **recursive function** as `f(...)` and all other parts above in `Q2()`. Note that

- **No need** to check the number's correctness from "input.txt" (downloaded from Canvas).
- **No need** to generate any output file. All the outputs are printed on the screen.

Example-1 (The input values are underlined):

```
The number from the input file is: 27
The result of f(n) is: 1277686868

Please input the start index and end index: 4 4
The selected number is: 6

This is a special number
The number of the factors being prime number(s) is: 2
```

Example-2 (The input values are underlined):

```
The number from the input file is: 15
The result of f(n) is: 470832

Please input the start index and end index: 2 3
The selected number is: 8

This is a not special number
```

Example-3 (The input values are underlined):

```
The number from the input file is: 30
The result of f(n) is: 2019485089

Please input the start index and end index: 0 9
The selected number is: 2019485089

This is a not special number
```

Example-4 (The input values are underlined):

```
The number from the input file is: 19
The result of f(n) is: 15994428

Please input the start index and end index: 6 7
The selected number is: 28

This is a special number
The number of the factors being prime number(s) is: 2
```

Q3: [Program Comprehension and Development] (25%)

Implement a class called `Robot`, which has four **private member variables**:

- `name[…]`: a `char` array, storing the robot's name.
- `power`: an `int` variable, storing the robot's remaining power value.
- `month`: an `int` variable, storing the month of the robot's last maintenance date.
- `day`: an `int` variable, storing the day of the robot's last maintenance date.

The `Robot` class also has the following **public member functions**:

- `getName()`: returns the robot's name.
- `getPower()`: returns the robot's remaining power value.
- `getMonth()`: returns the robot's last maintenance month.
- `getDay()`: returns the robot's last maintenance day.
- `set(char *n, int p, int m, int d)`: sets each member variable.

Part (a): Complete the class definition of `Robot` (Do **NOT** change the class name and do **NOT** add any extra members).

- Declare all the **member variables**.
- Robot's name is a **cstring** and contains **four** characters. Please determine the array size of `name[…]`, which should be **just** large enough to store the robot's name.

- Implement `getName()`, `getPower()`, `getMonth()` and `getDay()` **inside the class directly**.
- Implement `set(…)` **outside** the class.

Part (b): In `Q3()`, we provide the information for ten robots, including their names, remaining powers and latest maintenance dates. Please declare an array to store 10 `Robot` objects, and use the information provided above to initialize each `Robot` object.

Next, implement a **non-member** function, `sortRobots(Robot rArr[])`. It sorts an array of `Robot` objects by using Bubble sort according to:

- Maintenance date, with both **month** and **day**, in an **increasing** order, e.g.,
  - robot maintained on Oct. 15 is smaller than the robot maintained on Oct. 20, and
  - robot maintained on Sept. 15 is smaller than the robot maintained on Oct. 20.
- We ignore **year** and consider **month** and **day only** in this question.
- If two robots have the **same** maintenance date, further sort them based on their power values in an **increasing** order. We assume the powers of all the robots are different.

Develop the `sortRobots(…)` function, implement the Bubble sort algorithm and print the sorted robots, as shown in the example.

Part (c): According to the sorted order of these 10 robot objects in Part (b), we obtain a set of power values, e.g., 88 51 69 25 73 90 48 55 19 68 in the example. Then, please implement another **non-member** function, `findSeq (Robot rArr[])`, to find the the **longest continuous peak** sequence from this set of power values, i.e., the sequence is **first increasing** and **then decreasing**.

For the set of power values 88 51 69 25 73 90 48 55 19 68 in the example below:

- Because $25 < 73 < 90$ and $90 < 48$, the length of the sequence {25, 73, 90, 48} is four. We cannot find another continuous sequence longer than this one. So, {25, 73, 90, 48} is the one to be found.
- {25, 73, 90, 48, 19} is not considered because it is not **continuous** due to 55.

Implement the `findSeq (Robot rArr[])` function to print both the **length** and **each element** of the longest continuous peak sequence.

- We assume the powers of all the robot objects are different.
- If there exist multiple such sequences with the same length, print the **first** sequence.

- The peak sequence much contain both the **increasing** and **decreasing** two parts.
- If there is no continuous peak sequence, its length is 0 and leave the last line in the output to be empty.

Example (For the part of the sorted robots below, each column is separated by one '\t'):

```
Sorted robots (Name  Month-Day  Power):
R-03   1-1   88
R-04   3-1   51
R-01   5-1   69
R-02   7-20  25
R-09   7-20  73
R-05   8-15  90
R-06   9-15  48
R-07   10-15 55
R-10   10-20 19
R-08   10-20 68

The power values from Part (b):
88 51 69 25 73 90 48 55 19 68
Length of the longest continuous peak sequence is: 4
25 73 90 48
```

We provide another set of power values in ".cpp" to help you test your solution. If you use these power values, the sorted power values from Part (b) are 60 51 69 65 73 40 48 55 68 79. In this case, the length of the longest continuous peak sequence is 3 and the elements are 51 69 65.

Q4: [Program Development] (25%)

Write a program to read a sentence from the keyboard into a cstring using `str[100]`. The input contains **at least one** uppercase/lowercase English letter. In addition, it may also contain space(s) and/or some non-alphabet characters, e.g., '!', '6', '~', etc. You may use `strlen()` from <cstring>, but do not use other functions from this library in this question.

Part (a): Write in the `Q4()` to print the **longest word** in the input cstring. Two words are separated by a space. The **length** of one word is measured by the **number** of uppercase/lowercase **English letters only**, **without** the space(s) and non-alphabet characters.

- Print the **longest** word, i.e., the word that has the largest number of English letters. If there are multiple words having the **same** (**longest**) length, the first one should be printed.
- Non-English letters should **NOT** be counted as part of a word and should be ignored, e.g., "C++" is treated as word "C" of length 1, "!x+6-y" is treated as word "xy" of length 2, etc.
- When you print the longest word, print the English letter(s) only.

Part (b): We define a special feature of one word as follows: this word reads the **same forward** and **backward**, e.g., "ffttff" and "abcba". For the word obtained in Part (a), please write in `Q4()` to print its *smallest feature pattern* by adding **zero** or **several** characters at its end as follows.

- If this word already has the special feature stated above, **no** (**zero**) character needs to be added and its smallest feature pattern is itself. For example, *level* and *I* are such words.
- If this word does not have such a feature, its smallest feature pattern is obtained by adding the **smallest** number of extra characters at its end. For example, the smallest feature pattern of *come* is *comemoc*, by adding *moc* at the end of *come*.

Hint: You can reverse this word to determine the characters to be added. For example, the reversed word of *come* is *emoc*.

Example-1 (The input value is underlined):

```
Enter a string: I #c!ome# from CS######
The first longest word is: come
Its smallest feature pattern is: comemoc
```

Example-2 (The input value is underlined):

```
Enter a string: level of robot power is ten
The first longest word is: level
Its smallest feature pattern is: level
```

Example-3 (The input value is underlined):

```
Enter a string: <good afternoon>
The first longest word is: afternoon
Its smallest feature pattern is: afternoonretfa
```

--- END ---