How to be happy?

This is a question that everyone is looking for every day. Happiness is something everyone strives for, but not everyone knows how to achieve it.

First, happiness is not external but peace and serenity in the soul. When you practice living in the present, observing everything around you without judgment or comparison, you will feel relieved and at peace. Try sitting quietly, observing your breathing, and letting your mind rest in each moment; you will feel very happy.

And now, are you ready to receive new happiness?

The first happiness:
Let's write a function in C that takes a string. Print that string to the screen with the string "Happy " at the beginning of it. Libraries stdio.h, stdlib.h, string.h are allowed.
Function prototype:
`void makeHappiness(char input[])`

Example:
```
char input[] = "Tiger";
makeHappiness(input);
// "Happy Tiger" is printed to the screen (excluding the ").
```

The second happiness:
Implement a function in C to calculate the "Happiness score" of a given string. Happiness score is the sum of the ASCII codes of all vowels (a, e, i, o, u, A, E, I, O, U) in the string. The function will print the happiness score and a message indicating whether the score is "Happy" (score > 100) or "Unhappy" (score <= 100). If there are no vowels in the string, score is 0. Libraries stdio.h, stdlib.h, string.h are allowed.
Function prototype: `void calculateHappinessScore(char input[])`
Format of output: "`Happiness Score: <happiness_score> (<status>)`". (Do not include "). In which:

- `<happiness_score>` : is a non-negative integer
- `<status>` : is "Happy" or "Unhappy"

Example:
```
calculateHappinessScore("Hello, World!");
// Output: "Happiness Score: 323 (Happy)"
// e + o + o = 101 + 111 + 111 = 323
calculateHappinessScore("Sad");
// Output: "Happiness Score: 97 (Unhappy)"
// a = 97
```

The third happiness:
Write a function in C that takes a string consisting of non-negative integers separated by commas as input. The function will then find and return the "Happiest number" in the string. "Happy number" is a prime number, and the sum of its digits is a power of base 2. "Happiest number" is the "Happy number" with the largest sum of digits. If there is more than one "Happy number" with the largest sum of digits, the "Happiest number" will be the largest number. If there is no "Happy number", the "Happiest number" is 0. The libraries stdio.h, stdlib.h, math.h, stdbool.h, string.h are provided.
Function prototype: `int findHappinestNumber(char input[])`

Example:
```
char input[15] = "13,89,53,45,67";
int happiestNumber = findHappinestNumber(input);
// 13, 53 are prime numbers and the sum of the digits is a power of 2
// The sum of the digits of 53 (5+3=8) is greater than 13 (1+3=4), so 53 is chosen as the Happiest number.
```

The fourth happiness:
Sometimes, expressing words or arranging words in a sentence appropriately will make the listener feel happier and more peaceful. Write a C function that takes a string, rearranges the string's characters to get ***the highest "Happiest score"*** and prints the result to the screen.

You have the right to transpose all characters in the string. A permutation would be equivalent to an array of "Happiness scores". The "Happiness score" in this sentence is calculated as the sum of the ASCII codes of adjacent vowels (a, e, i, o, u, A, E, I, O, U) (vowel clusters). For example, "aebcdef" will be equivalent to a two-element array of "Happiness scores" corresponding to "ae" and "e". "Happiness score" can be "Happy number" or not. "Happy number" is a prime number and the sum of its digits is a power of base 2. "Happiest score" ***must*** be a "Happy number" and have the greatest sum of digits. For each permutation of the string there will be a corresponding "Happiest score". You need to find the largest "Happiest score" and the vowel cluster corresponding to this number across all permutations of the string. If there are no vowels in the input string, the largest "Happiest score" is 0.

In case there is more than one "Happiest score" with the same value, the vowel cluster has a shorter length and has more vowels (For example: "aia" has 2 vowels, 'a' and 'i') will prioritized. In the case of the same "Happiest score", the same length and number of vowels in the vowel cluster, the different first character of the vowel cluster with the larger ASCII code will be chosen. The libraries stdio.h, stdlib.h, math.h, stdbool.h, string.h are provided.

Function prototype: `void maximizeHappiestScore(char input[])`
Format of output: "`Happiest String: <happiest_string>\nHappiest Score: <happiness_score> (<status>)`" (Does not include "). In which:

- <happiest_string>: Is a vowel cluster arranged in ascending order according to the ASCII code corresponding to the Happiest score found.
- <happiness_score>: is a non-negative integer
- <status>: is "Happy" or "Unhappy"

For example:

```
char input[11] = "i love you";
maximizeHappiestScore(input);
/*
* The string "i love you" will correspond to the array Happiness score {"i", "o", "e", "ou"} ⇒ {105, 111, 101, 228}. In this array, there are Happy numbers {101}. Therefore, the Happiest score corresponding to the string "i love you" will be 101.
* Do the same process with all permutations of the string "i love you". For example: "i love yuo", "i love tuyo", "i love uoy", … (There are a total of 10! permutations with the string "i love you")
* Finally, you will find the combination of vowels with the highest Happiest score: e. Happiest score: e = 101
*/
// Output: "Happiest String: e\nHappiest Score: 101 (Happy)"
```

The fifth happiness:

Indicate the Group and Member structures as below.

```
typedef struct {
     Member* members;
     int numMembers;
     int happinessLevel;
} Group;

typedef struct {
     char* name;
     int happinessLevel;
} Member;
```

Given an array of members and a standard "Happiness level", write a function in C to group members into as many groups as possible so that the group's "Happiness level" is greater than or equal to the standard "Happiness level" (meets standard). A group has at least 1 member. Know that the "Happiness level" of a group will be the sum of the "Happiness level" of the members in that group. There may exist a group whose "Happiness level" is less than the standard "Happiness level". After grouping, the names of members of groups with a qualified "Happiness level" will be added with the phrase "Happy " at the beginning of the name. For example, "Andy" will become "Happy Andy". The result of this function will be an array containing **only** groups satisfying "Happiness level" standards, regardless of order. If there is

no member, the function returns NULL. The libraries stdio.h, stdlib.h, math.h, stdbool.h, string.h are provided.

Function prototype:
`Group* splitGroups(Member* const members, int numMembers, int standardLevel)`

For example:

```
Member members[] = {
     {"Alice", 5},
     {"Bob", 7},
     {"Charlie", 3},
     {"David", 8},
     {"Eve", 6},
     {"Frank", 4},
     {"Grace", 9},
     {"Henry", 2},
     {"Ivan", 1},
};
Group* groups = splitGroups(members, 9, 6);
/*
* A correct way to divide groups:
* Group 0: [{"Happy Grace", 9}]
* Group 1: [{"Happy David", 8}]
* Group 2: [{"Happy Bob", 7}]
* Group 3: [{"Happy Eve", 6}]
* Group 4: [{"Happy Alice", 5}, {"Happy Ivan", 1}]
* Group 5: [{"Happy Frank", 4}, {"Happy Henry", 2}]
*/
```

Although it is difficult to evaluate happiness, the results of happy things are explicit.

- 1st happiness: 1 point
- 2nd and 3rd happiness: 2 points/question
- 4th and 5th happiness: 2.5 points/question

To be happy, you must submit your work on the LMS system. Each of you has **6 opportunities** to submit your answers to find happiness. However, only **final submission** counts, so know how to seize the opportunity to find happiness. The submission time for finding happiness is 10 minutes. Note that you must complete all assignments before submitting them to the system.

Submission system opening time: **Monday, May 27, 2024**
Submission system closing time: **Tuesday, June 4, 2024**
You can submit your work anytime while the submission system is open.

Handling fraud:

This assignment must be completed individually. If cheating is detected, including copying someone else's work, buying, selling assignments, or other cases of cheating without limitation, the person committing the cheating act and related individuals will receive a score of 0 (Zero) for this assignment and other disciplinary measures according to the regulations of the Faculty and University.

Troubleshooting:

Questions about content related to the assignment will be answered on the subject's general forum. Individual emails with questions about the assignment content will not be answered.

In addition to questions about the assignment content, other requests/questions can be emailed to:

- Lecturer Nguyen Quang Duc (nqduc@hcmut.edu.vn)

Version: 1.0