Mini-projects Course 20192

Select one of the following problems, write an assembly program.

Projects

(1). Create a program to input a text line from the keyboard and test if it is a palindrome. For example: "**abc1**2**1cba**" is a palindrome. Store all palindromes which the user typed into the **memory**, to make sure that the user does not duplicate palindromes.

(2). Find all prime numbers (such as 2, 3, 5, 7, etc) in a range from the integer N to the integer M. N, M is entered from the keyboard.

(3*). Create a program to convert from number to text, in English or Vietnamese (choice 1 of 2). The number in range from 0 to 999 999 999

For example:

Input: 1432

Output: one thousand four hundred and thirty-two

(4). Create a program to:

- Input an array of integers from the keyboard.

- Find the maximum element of the array.

- Calculate the number of elements in the range of (m, M). Range m, M are inputted from the keyboard.

(5). Write a program to get decimal numbers, display those numbers in binary and hexadecimal.

(6) Given an array of word elements and the number of elements, write a procedure to find the pair of adjacent elements that has the largest product and return that product.

Example: For inputArray = [3, 6, -2, -5, 7, 3], the output should be the product of 7 and 3 (21)

(7) Some people are standing in a row in a park. There are trees between them which cannot be moved. Your task is to rearrange the people by their heights in a non-descending order without moving the trees. People can be very tall!

Example: For a = [-1, 150, 190, 170, -1, -1, 160, 180], the output should be sortByHeight(a) = [-1, 150, 160, 170, -1, -1, 180, 190].

(8). Write a program to:

You must create 1 variable (number students) and 2 arrays (student name, mark) to store the input data

- Input the number of students in class.
- Input the name of students in class, Math mark (0 →10)
- Sort students due to their mark.
- Print list of students to screen

(9). Write a program to:

Assume that you already have 1 variable (number students) and 2 arrays (student name, mark) in memory

- Read in the number of students in the class.
- Read information about each student, including: Name, Math mark.
- List the names of all students who have not passed the Math exam.

(10*) Write a program that gets an integer i from the user and creates the table shown below on the screen (example inputs provided). Subroutines are required for power, square, and hexadecimal (in 32 bit arithmetic, attend to overflowed results). Hint: Hexadecimal can be done with shifts and masks because the size is 32 bits.

i	power(2,i)	square(i)	Hexadecimal(i)
10	1024	100	0xA
7	128	49	0x7
16	65536	256	0x10

(11) Programming an application to convert names from LastName-FirstName to FirstName-LastName.

1. Input name of 2 students with space character, for example “Vu Thi XYZ”. Store them into the memory.
2. Change them from LastName-FirstName to FirstName-LastName, for example “XYZ Vu Thi”

3. Print names to the screen.

(12) Parsing an ASCII string to binary number

Write a function that converts a string of ASCII digits into a 32-bit integer. The function will receive as an argument the starting address of the string and must return a 32-bit integer containing the integer value of the string. Assume that the string is an ASCIIZ string, i.e., ends with the null character (ASCII code 0). You do not need to check for errors in the string, i.e., you may assume the string contains only characters '0' through '9' (i.e., their corresponding ASCII codes), and will not represent a negative number or a non-decimal value or too large a number. For example, a_to_i called with the argument "12345" will return the integer 12345. **atoi()**

(13) Ticket numbers usually consist of an even number of digits. A ticket number is considered lucky if the sum of the first half of the digits is equal to the sum of the second half. Given a ticket number n, determine if it's lucky or not.

Example

For n = 1230, the output should be isLucky(n) = true;

For n = 239017, the output should be isLucky(n) = false.

(14). Given two strings, find the number of common characters between them.

Example: For s1 = "aabcc" and s2 = "adcaa", the output should be commonCharacterCount(s1, s2) = 3. Strings have 3 common characters - 2 "a"s and 1 "c".

(15) You are given an array of integers. On each move you are allowed to increase exactly one of its elements by one. Find the minimal number of moves required to obtain a strictly increasing sequence from the input.

Example: For inputArray = [1, 1, 1], the output should be arrayChange(inputArray) = 3. [1,2,3]

The minimal number of moves needed to obtain a strictly increasing sequence from inputArray. It's guaranteed that for the given test cases the answer always fits signed 32-bit integer type.

(16) Given a sequence of integers as an array, determine whether it is possible to obtain a strictly increasing sequence by **removing no more than one element** from the array.

Note: sequence a0, a1, ..., an is considered to be strictly increasing if a0 < a1 < ... < an. Sequences containing only one element are also considered to be strictly increasing.

Example:

- For sequence = [1, 3, 2, 1], the output should be almostIncreasingSequence(sequence) = false. There is no one element in this array that can be removed in order to get a strictly increasing sequence.

- For sequence = [1, 3, 2], the output should be almostIncreasingSequence(sequence) = true. You can remove 3 from the array to get the strictly increasing sequence [1, 2]. Alternately, you can remove 2 to get the strictly increasing sequence [1, 3].

(17) Write a program that inputs a string. Extract number characters and show to screen in inverse order using stack.

(18) Two arrays are called similar if one can be obtained from another by swapping at most one pair of elements in one of the arrays.

Given two arrays a and b, check whether they are similar. Example:

- For a = [1, 2, 3] and b = [1, 2, 3], the output should be areSimilar(a, b) = true. The arrays are equal, no need to swap any elements.

- For a = [1, 2, 3] and b = [2, 1, 3], the output should be areSimilar(a, b) = true. We can obtain b from a by swapping 2 and 1 in b.

- For a = [1, 2, 2] and b = [2, 1, 1], the output should be areSimilar(a, b) = false. Any swap of any two elements either in a or in b won't make a and b equal.

(19) Write a program that input some variable names. Check if variable names consist only of English letters, digits and underscores and they can't start with a digit.

Example

- For `name = "var_1__Int"`, the output should be `variableName(name) = true`;
- For `name = "qq-q"`, the output should be `variableName(name) = false`;
- For `name = "2w2"`, the output should be `variableName(name) = false`.

(20) Given a string which consists of lower alphabetic characters (a-z), count the number of different characters in it.

Example: For `s = "cabca"`, the output should be `differentSymbolsNaive(s) = 3`.

There are `3` different characters `a`, `b` and `c`.

(21) Let's define the *digit degree* of some positive integer as the number of times we need to replace this number with the sum of its digits until we get to a one digit number. Given an integer, find its digit degree.

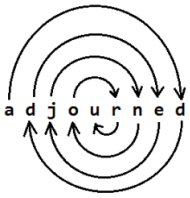
Example

- For `n = 5`, the output should be `digitDegree(n) = 0`;
- For `n = 100`, the output should be `digitDegree(n) = 1`. `1 + 0 + 0 = 1`.
- For `n = 91`, the output should be `digitDegree(n) = 2`. `9 + 1 = 10` -> `1 + 0 = 1`.

(22) Cyclone Word (challenge)

Cyclone words are English words that have a sequence of characters in alphabetical order when following a cyclic pattern.

Example:



Write a function to determine whether a word passed into a function is a cyclone word. You can assume that the word is made of only alphabetic characters, and is separated by whitespace.

is_cyclone_phrase("adjourned") # => True

is_cyclone_phrase("settled") # => False

Cách thực hiện:

- Mỗi sinh viên chọn một bài
- Cách phân chia nhóm do SV tự sắp xếp và gửi giáo viên
- Các bài tập của mỗi nhóm sẽ được gán ngẫu nhiên cho các nhóm
- Thời gian làm bài: tuần 9, 10. Sinh viên làm việc ở nhà
- Tuần kiểm tra: tuần 11

Kết quả thực hiện:

- Viết báo cáo trình bày
  - phân tích cách thực hiện
  - ý nghĩa của các thanh ghi được sử dụng
  - ý nghĩa của các chương trình con nếu có
- Mã nguồn chương trình
  - Có chú thích trong mã nguồn. Ví dụ

```
#-----------------------------------------------------------------------
# @brief      Kiem tra hieu ung cua mot scene bang cach polling
# @param[in]  Scene_Ptr bien toan cuc, co gia tri la dia chi cua
#                  Scene vua duoc thiet lap, dia chi cua Scene 1,2,3,4
# @param[in]  Scene_Len bien toan cuc, cho biet do dai
# @return     $v0     Thanh ghi chua ma loi
# @note          Phai goi ham SetScene truoc
#-----------------------------------------------------------------------
.ent Test_Scene
Test_Scene:
        jal NextFrame
        nop
```

```
        jal ShowScene
        nop
        j   Test_Scene
.end Test_Scene
```

Cách kiểm tra

- Từng nhóm 2 sinh viên gặp giáo viên để kiểm tra.
- Khi gặp giáo viên kiểm tra, các SV có thể dùng máy tính laptop, nếu không có thể mời giáo viên tới bàn máy tính tại Lab.
- Mỗi SV sẽ trình bày trả lời các câu hỏi của giáo viên về 1 (trong 2 bài tập của nhóm).
- Nội dung chính để kiểm tra:

1. Chạy chương trình và cho kết quả đúng
2. Hạn chế được các lỗi thao tác nhập liệu của người dùng (ví dụ nếu người dùng cố tính nhập giá trị số nguyên là hello)
3. Hạn chế được các số quá lớn, ngoài phạm vi chương trình (ví dụ, tính giai thừa của 1 tỷ)
4. **Hiểu được ý nghĩa của các lệnh sử dụng trong bài**
5. **Trả lời được các câu hỏi lý thuyết ứng với các lệnh trong chương trình (khuôn dạng của lệnh này là gì? Lệnh này mất bao nhiêu chu kì để thực hiện…)**
6. Chỉnh sửa trực tiếp được chương trình.
7. Mã nguồn có chú thích đầy đủ, rõ ràng.