VIII ОЛИМПИАДА ШКОЛЬНИКОВ ПО ИНФОРМАТИКЕ И КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

9-10 КЛАСС

Задача 1. Подбор пароля

В результате анализа подсистемы защиты удалось выяснить, что подтверждение имени пользователя, работающего за компьютером, осуществляется с использованием паролей. При проверке введенный пароль подвергается преобразованию при помощи функции, исходный код которой приведен ниже (см. таблицу 1). Результат работы функции сверяется с эталонами, хранимыми в базе данных, для принятия решения о том, верен пароль или нет. Приведите пароль, который пройдет описанную проверку, если известно, что пароль «АУТЕНТИФИКАЦИЯ» верен.

Таблица 1

C

```
int HASH(char *text)
{
  int k=0, H;
  char temp[4];
  for (int i=0; (i<strlen(text))&&(k<4); i++)
{
    if ((i+1)%2==1)
{
temp[k]=text[i]; k++;
}
  }
  if (k<4)
  {
for(;k<4;k++)
temp[k]='F';
  }
  H = ((temp[0]-temp[3])&255)*256;
  H += (temp[1]-temp[2])&255;
return H;
}
```

Pascal

```
function HASH (text: string):Integer;
var
 i, k, HA : Integer;
 temp: array [1..4] of char;
begin
 k := 1; i := 1;
 while ((i <= Length (text)) and (k <= 4)) do
 begin
   if (i mod 2 = 1) then
   begin
     temp[k] := text[i];k := k+1;
   end;
   i := i+1;
 end;
 if (k <= 4) then
 begin
   while k <= 4 do
   begin
     temp[k] := 'F'; k := k+1;
   end;
 end;
 HA := ((ord(temp[1])-ord(temp[4])) and 255)*256;
 HA := HA + ((ord(temp[2])-ord(temp[3])) and 255);
 HASH := HA;
end;
```

Решение

Задача решается аналогично задаче первого варианта. Но для решения необходимо учесть, что:

- первоначально из пароля выбираются символы, стоящие на нечетных местах;
- если количество выбранных символом меньше 4, то в конец строки temp дописывается «F»;

старший байт образа равен разности первого и четвертого символа строки temp, а младший – разности второго и третьего символа;

- искомое значение образа равно $63493_{10} = 1111100000000101_2$.

Обозначив первый и третий символ пароля, как P_1 и P_2, составим следующие уравнения:
P_1-70=248 (mod 256) и P_2-70=5 (mod 256).

Таким образом, пароль >*K* (где * – любой символ) верен.

Ответ: >*K*

<u>Задача 2.</u> Доступ

Специалисты по информационной безопасности, проанализировав компьютерную систему, пришли к следующим выводам:

1. В системе хранятся файлы F_1, F_2, F_3;
2. Система имеет в своем составе набор программ S_0, S_1, …, S_6;
3. Удалось определить права, которыми обладают программы. Права доступа представлены в виде таблицы доступов.

Выясните, возможно ли чтение данных программой S_0из файла F_1? Ответ обосновать.

	F_1	F_2	F_3	S_0	S_1	S_2	S_3	S_4	S_5	S_6
S_0										
S_1				g						
S_2								t		
S_3					t					
S_4									t	
S_5	r									
S_6						g	t			

Комментарий

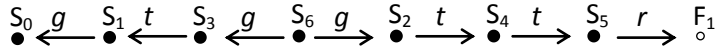
Каждая строка таблицы доступов описывает права одной программы в системе. Каждый элемент строки описывает, какими правами обладает программа по отношения к элементу системы, которым помечен соответствующий столбец. Например, в приведенной таблице программа S_5имеет право на чтение файла F_1. В общем случае, программа может иметь несколько прав к одному и тому же элементу системы.

Право r (*read*) показывает, что программа может обратиться и считать данные, относящиеся к элементу. Права, обозначенные как *t* (*take*) и *g* (*grant*), являются соответственно правом брать право и давать право. Обладая этими правами, программы могут изменять набор прав доступа согласно правилам, приведенным в таблице.

Название правила	**Состояние элементов таблицы доступов до применения правила**	**Состояние элементов таблицы доступов после применение правила**
Take	O S_1 S_2 S_1 β S_2 t S_i – программа; O – файл или программа; β – любое право доступа	O S_1 S_2 S_1 β S_2 β t
Grant	O S_1 S_2 S_1 S_2 β g S_i – программа; O – файл или программа; β – любое право доступа	O S_1 S_2 S_1 β S_2 β g
Create	S S S – программа;	O S S β S – программа; O – файл или программа;

Решение

Аналогично решению задачи варианта 1 строим граф доступов.



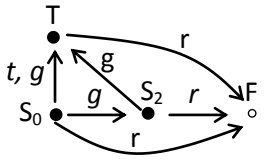
Используем следующие преобразования графа доступа:

1. программа S_4 преобразовывает граф по правилу *take;*

2. программа S_2 преобразовывает граф по правилу *take;*

3. программа S_1 преобразовывает граф по правилу *take;*

4. программа S_3 преобразовывает граф по правилу *take;*
5. программа S_6 преобразовывает граф по правилу *grant;*
6. программа S_0 создает программу T по правилу *create*;
7. программа S_0 преобразовывает граф по правилу *grant*;
8. программа S_2 преобразовывает граф по правилу *grant*;
9. программа S_0 преобразовывает граф по правилу *take*.



В итоге, из полученного графа можно сделать вывод, что чтение данных из файла *F* программой S_0 возможно.

<u>Задача 3.</u> Сеть

В офисе 15 компьютеров объединены в сеть, посредством 22 проводных соединений. В целях обеспечения конфиденциальности информации было принято решение о введении дополнительных паролей для следующих групп, которые были построены по следующим принципам:

1. Если в последовательности соединенных компьютеров отправленная информация может вернуться отправителю, при этом не проходя два раза по одному и тому же каналу связи, то эти компьютеры объединяют в одну группу и используют один дополнительный пароль.
2. Компьютер может входит одновременно в несколько групп.
3. Если компьютеры группы содержатся в объединении двух и более групп, то этой группе не выделяется дополнительный пароль.
4. Если компьютер не входит ни в одну из таких групп, ему не выделяется дополнительный пароль.

Какое минимальное количество различных дополнительных паролей будет достаточно для осуществления такой связи?

Решение

Представим компьютерную сеть в виде графа, в компьютеры – это вершины, а рёбра – каналы связи. Поскольку все компьютеры имеют возможность обмениваться информацией между собой, то граф сети должен быть связен. Согласно условию в графе 15 вершин. Минимальное количество рёбер, необходимое для их соединения и образования связного графа – 14. (если надо можно доказать). Предположим, что соединили компьютеры с номерами i и j. Между ними в графе уже существовал путь, добавление ребра образовало замкнутый путь – цикл. Поэтому всякий раз добавление в граф одного из оставшихся 8 рёбер будет приводить к тому, что будет образовываться новая группа. Следовательно, при добавлении очередного ребра к графу может возникнуть необходимость увеличения числа секретных ключей на 1. Число ключей равное 8 гарантированно обеспечит связь при любой конфигурации сети.

Ответ: 8 ключей.

<u>Задача 4.</u> Путаница

Вася совсем запутался и не может понять, что делает функция g, которую написал Петя. Объясните ему, каким образом она это делает (единственное, что он точно знает так это то, что на вход g Петя подает структуру p, у которой поле заполнено различными целыми числами от 0 до N-1).

```
#define N 10
                 struct P{int v[N];};     P a(P p){P r; int i=N;
      while(i-->0)r.v[p.v[i]]=i;                 return r;}P g(P p,int k)
{P r=p;while(k){if(k&1)r=a(p);                       p=a(p);k>>=1;}return r;}
```

Решение

Восстановим удобочитаемую структуру программы.

```
(1) #define N 10
(2) struct P { int v[N]; };

(3) P a(P p){
(4)  P r;
(5)  int i=N;
(6)  while(i-->0)
(7)    r.v[p.v[i]]=i;
(8)  return r;
(9)}

(10) P g(P p, int k)
(11) {
(12)  P r=p;
(13)  while(k){
(14)    if(k&1)
(15)     r=a(p);
(16)   p=a(p);
(17)    k>>=1;
(18)  }
(19)  return r;
(20)}
```

По аналогии с решением задачи 4 можно показать, что gвозводит перестановку p в степень k методом бинарного возведения в степень.

<u>Задача 5.</u> Фильтр

Вася предоставил соседу Пете доступ к своей домашней Wi-Fi сети и сообщил ему ключ. Хитрый Петя захотел узнать имена всех учётных записей, которыми пользуется Вася на сайтах знакомств. Для перехвата данных он решил использовать специальную программу (сниффер). Петя очень ленив и не хочет проверять все перехваченные данные вручную, поэтому он воспользовался возможностями сниффера по анализу содержимого html-страниц с применением регулярных выражений. Помогите Пете написать регулярное выражение, позволяющее выделить искомый логин. Код типовой страницы авторизации имеет следующий вид:

```
$text = '
<html>
<head>
<title>Наш сайт знакомств лучший</title>
</head>
<body>
<form>
<input type="hidden" name="login"
value="vasyaisthebestofthebest@mail.ru "/>
<input type="text" id="user56789" name="login"
value="vasyaisthebestofthebest@mail.ru" class="loginstring"/>
<input type="hidden" name="password" value="bestman12"/>
```

```
<input type="password"  name="password" class="passwordstring"/>
<input type="submit" name="action" value="Войти на сайт!" />
</form>
</body>
</html>';
```

Комментарий

Регулярные выражения предоставляют возможности для описания подстрок определенного вида. Для формирования регулярного выражения используются следующие элементы:
1. символ – например, «a»
2. любой символ – обозначается «.»
3. пробельный символ – обозначается «\s»
4. диапазон символов – обозначается «[]».Например:
[abc] – любая из букв a, b, c
[a-z0-9] – любая из малых букв латинского алфавита и цифра
5. отрицание диапазона:
[^a-z5] – не маленькая буква латинского алфавита и цифра 5

Для указания количества вхождений используются кванторы:
«?» означает «0 или 1шт.»
«+» означает «>= 1шт.», причем берется как можно большее количество символов
«+?» означает «>= 1шт.», причем берется как можно меньшее количество символов
«*» означает «>= 0шт.», причем берется как можно большее количество символов
«*?» означает «>= 0шт.», причем берется как можно меньшее количество символов
«{5}» означает «5 шт.»
«{5, 8}» означает «от 5 до 8 шт.»
Например, «a*» – любое количество идущих подряд символов «a».

Для извлечения подстроки используются круглые скобки, например, выражение: ([0-9]+)\s+([0-9]+) сохраняет из строки «мои числа 45 567 234 56» числа: «45» и «567».

Решение

Строка, содержащая пароль, содержит открывающийся тэг <*input*, атрибуты *type* с опцией *hidden*и атрибут *name* с опцией *password*. Эти атрибуты соединены одним или несколькими пробельными символами. Таким образом, регулярное выражение должно начинаться следующим образом:
<input\s+type="hidden"\s+name="password"

Далее в атрибуте *value* храниться пароль, который записан в кавычках. Пароль может содержать любые символы, кроме кавычек, а значит, выражение, позволяющее его получить, выглядит следующим образом:
value="([^"]*)".

Соединяя результаты, полученные в первом и втором параграфах, при помощи любого количества пробельных символов, получаем искомое выражение:
<input\s+type="hidden"\s+name="password"\s+value="([^"]*)"

Ответ: <input\s+type="hidden"\s+name="password"\s+value="([^"]*)"