



XII МЕЖРЕГИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ПО ИНФОРМАТИКЕ И КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ 2017 год



ВАРИАНТ 1

Задача 1. Хеш-значения

Политика безопасности ОС не позволяет задавать для учетных записей пользователей пароли, совпадающие с их именами. Для этого перед добавлением нового пользователя в базу вызывается функция `CheckUser()`. При ее успешном выполнении (возвращаемое значение = 0) в базу добавляется новая запись, содержащая имя учетной записи пользователя и хеш-значение пароля, полученное с помощью функции `Hash()`.

Си	Паскаль
<pre>int CheckUser(char* username, char* password) { for (int i=0; i<strlen(username); i++) { if ((username[i] >= 0x30 && username[i] <= 0x39) (username[i] >= 0x41 && username[i] <= 0x5A) (username[i] >= 0x61 && username[i] <= 0x7A)) continue; else return 1; } for(int i=0; i<strlen(password); i++) { if (password[i] >= 0x23 && password[i] <= 0x7D) continue; else return 1; } if (strcmp(username, password) != 0) return 0; else return 1; }</pre>	<pre>function CheckUser(var username: string; password: string) : integer; var i:integer; begin for i:=1 to Length(username) do begin if(((username[i] >= #48) and (username[i] <= #57)) or ((username[i] >= #65) and (username[i] <= #90)) or ((username[i] >= #97) and (username[i] <= #122))) then continue else begin Result := 1; Exit; end; end; for i:=1 to Length(password) do if((password[i] >= #35) and (password[i] <= #125)) then continue else begin Result := 1; Exit; end; end; if(UpperCase(username) <> UpperCase(password)) then Result := 0 else Result := 1; end;</pre>
<pre>const char letters[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890"; char* Hash(char* str) {</pre>	<pre>function Hash(str:string[255]): string[20]; var Nstr,hsize:integer; var i,p_hash,p_pow,p:word; var buf:string[255]; var size:integer;</pre>

```

const int p=31;
const int Nstr=32;
const int Hsize=Nstr/2;
unsigned short int hash=0, p_pow=1;
char buf[Nstr+1]={0};
int size=0;
char *res=new char[Hsize+1];
while (str[size]!='\0' && size<Nstr)
{
    buf[size]=str[size];
    size++;
}

for (int i=size; i<Nstr; i++)
{
    buf[i]='A'+(i-size);
}
for (int i=0; i<Nstr; i+=2)
{
    hash+=(buf[i]-'a'+1)*p_pow;
    res[i/2]=letters[hash%strlen(letters)];
    p_pow*=p;
}
res[Hsize]='\0';
return res;
}

```

```

var letters:string[100];
var res:string[32];
begin
    letters:='abcdefghijklmnopqrstuvwxyz
    ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890';
    p:=31;
    Nstr:=32;
    Hsize:=Nstr div 2;
    p_hash:=0;
    p_pow:=1;
    size:=1;
    while ( (size<=length(str)) and
            (size<=Nstr)) do
        begin
            buf:=buf+str[size];
            size:=size+1;
        end;
    for i:=size to Nstr+1 do
        buf:= buf+chr(ord('A')+(i-size));
    i:=1;
    repeat
        p_hash:=p_hash+(ord(buf[i]) -
            ord('a')+1)*p_pow;
        res:=res+letters[p_hash mod
            length(letters)+1];
        p_pow:=p_pow*p;
        i:=i+2;
    until i>Nstr;
    Hash:=res;
end;

```

В базе пользователей присутствуют следующие записи:

admin	rUZxHUUfw9oJSFNm
user	vYhkB2klurVYYPtq
operator	pS8HIUqmrJ60ZOrk
manager	ng3JfGZ0TQzmCtS5
root	sQUsDRwnsAf90HN0

В ходе проверки администратор выявил случаи возможного нарушения политики безопасности.

Определите:

- пользователей, для которых были обнаружены нарушения;
- причины возникновения нарушений.

Задача 2. Секретное сообщение

В исполняемый файл PROG.EXE было внедрено секретное текстовое сообщение. При этом сам файл корректно выполняет все функции. Известно, что для того, чтобы отметить место внедрения информации, нарушитель использовал 1-байтную метку, после которой идет сообщение размером до 10 байт:

Метка (1 байт)	Сообщение (10 байт)
----------------	---------------------

Какое сообщение было внедрено в файл?

К задаче прилагается: исполняемый файл PROG.EXE.

Задача 3. Антивирус

Для выявления вредоносного кода некоторым антивирусом применяется только сигнатурный метод анализа, позволяющий выполнять поиск известных сигнатур в файле путем побайтового сравнения. Файл считается вредоносным при наличии в нем участка данных, точно совпадающего с одной из сигнатур. Реализация поиска сигнатур описана в функции `CheckFile()`, которая возвращает `TRUE` при отсутствии сигнатур в файле и `FALSE` в противном случае.

Си	Паскаль
<pre> /* ВХОДНЫЕ ПАРАМЕТРЫ: * FNAME - имя анализируемого файла * SIGNATURE - сигнатура (массив байтов) * SIZE - размер сигнатуры в байтах * * ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ: * TRUE - файл не содержит сигнатуры * FALSE - файл содержит хотя бы одну * сигнатуру */ bool CheckFile(char* fname, char* signature, int size) { FILE *fin=NULL; char *buf=NULL; int read, check_count=0; buf=new char[size]; fin=fopen(fname, "rb"); if (!fin) return false; while(!feof(fin)) { read=fread(buf, 1, size, fin); if(read!=size) break; check_count=0; for (int j=0; j<size; j++) { if (buf[j]==signature[j]) check_count++; else break; } if (check_count==size) return false; } return true; } </pre>	<pre> /* ВХОДНЫЕ ПАРАМЕТРЫ: * FNAME - имя анализируемого файла * SIGNATURE - сигнатура (массив байтов) * SIZE - размер сигнатуры в байтах * * ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ: * TRUE - файл не содержит сигнатуры * FALSE - файл содержит хотя бы одну * сигнатуру */ function CheckFile (fname, signature: string; size: integer): boolean; var fin: file; ch: char; count_read, check_count, i: integer; buf: string; begin assign(fin, fname); reset(fin); while true do begin buf:=''; count_read:=0; for i:=1 to size do begin if (not eof(fin)) then begin read(fin, ch); buf:=buf+ch; count_read:=count_read+1; end; end; if count_read <> size then break; check_count:=0; for i:=1 to size do begin if(buf[i]=signature[i]) then check_count:=check_count+1 else break; end; if check_count=size then begin CheckFile:=false; Exit; end; end; close(fin); CheckFile:=true; end; end; end; </pre>

Какое количество файлов размером 7 байт, состоящих только из цифр от 0 (код 0x30) до 9 (код 0x39) включительно, может содержать трех-байтовую сигнатуру «0x313231» хотя бы один раз, но успешно проходить проверку антивирусом? Ответ обоснуйте.

Задача 4. Архив

Школьник скачал с некоторого Интернет-ресурса архив PROGS.RAR, который, согласно приведенному на сайте описанию, содержит пакет простых утилит (каждой утилите соответствует ровно один исполняемый файл формата .EXE). После распаковки архива школьник обнаружил, что часть файлов из архива зашифрована методом «двоичного гаммирования», т.е. путем выполнения операции «побитового исключающего ИЛИ» между байтами исходного файла и байтами, полученными циклическим повторением последовательности двух байтов ключа «0xB69D».

Зашифрованные файлы не запускаются, а при попытке запуска незашифрованных файлов, некоторые из них блокируются антивирусом из-за наличия в них подозрительной сигнатуры «0x0A0B0C0F».

Помогите школьнику получить из архива максимальное количество программ, которыми он сможет воспользоваться, не отключая антивирус.

К задаче прилагается: архив PROGS.RAR, скаченный школьником с Интернет-ресурса.

Задача 5. Контроль версий

В системе развернута инкрементная система контроля версий, хранящая исходные значения контролируемых файлов и их изменения в виде контрольных точек, по которым возможно восстановить текущее содержимое файлов.

Для файла config.txt в системе сохранено 5 контрольных точек:

- 1) 35 57 38 64 39 6F 3A 77 3B 73 3C 37
- 2) D3 54 D4 52 D5 55 D6 45 D7 22 D8 00
- 3) DA 35
- 4) D8 0D D9 0A DA 75 DB 73 DC 65 DD 72 DE 3D DF 22 E0 72 E1 6F E2 6F E3
74 E4 22
- 5) 96 38 98 32 C2 38 C4 31

Содержимое файла после 3-й контрольной точки приведено (config.txt.backup.3). Восстановите содержимое файла после 5-й контрольной точки.

К задаче прилагается: исходный файл CONFIG.TXT и файл после третьей контрольной точки CONFIG.TXT.BACKUP.3.
