Odpovědi pište na zvláštní odpovědní list s vaším jménem a fotografií. Pokud budete odevzdávat více než jeden list s řešením, tak se na 2. a další listy nezapomeňte podepsat. Do zápatí všech listů vždy napište i/N (kde i je číslo listu, N je celkový počet odevzdaných listů).

```
00000000: 2E 73 6E 64 00 00 00 18|00 00 00 93 00 00 00 02  | .snd.......“....
00000010: 00 00 13 88 00 00 00 01|F2 06 22 17 1F 18 21 09  | ........ň."...!.
00000020: E2 E5 E3 E5 E3 E6 13 1E|1A 1E 18 20 FE DF E7 E1  | âÍăÍăć..... ţßçá
00000030: E7 E1 EE 1B 1C 1C 1C 1A|1D F3 E0 E7 E1 E7 DF F9  | çáî.......óŕçáçßů
00000040: 20 19 1D 1A 1D 17 E9 E2|E5 E2 E6 E0 05 20 19 1E  |  .....éâÍâćŕ. ..
00000050: 1B 1C 19 EE E0 E6 E2 E6|E1 EE 19 1D 1B 1D 1A 20  | ...îŕćâćáî.....
00000060: 07 E0 E6 E2 E6 E1 F1 07|04 07 01 FA FA FB FB FB  | .ŕćâćáń....úúűűű
00000070: FA FC FB FF 06 05 05 04|06 06 04 06 02 FA FB FB  | úüű˙.........úűű
00000080: FB FA FA FA FA FB 03 06|05 05 06 05 06 05 06 03  | űúúúúű..........
00000090: FA FB FA FB FA FB FA FB|F9 03 05 05 05 05 06 06  | úűúűúűúűůů.......
000000A0: 04 05 03 FC FB FB FA FB|FA FB FA                 | ...üűűúűúűú
```

Společná část pro otázky označené X

Předpokládejte, že jsme si pomocí hex vieweru zobrazili kompletní obsah souboru `x.au`, o kterém víme, že je ve zvukovém formátu `.AU` – viz výpis výše v záhlaví tohoto zadání (popis formátu souborů `.AU` je **v příloze**).

Otázka č. 1 (X)

Předpokládejte nějaký běžný pevný disk, který je naformátovaný nějakým běžným souborovým systémem. Do souborového systému uložíme 4 soubory `a.au`, `b.au`, `c.au`, `d.au`, kde do každého z nich nakopírujeme právě obsah výše uvedeného souboru `x.au` (tj. obsah všech 4 nových souborů bude stejný). O kolik se uložením těchto souborů na disk zmenší celkové volné místo na disku (použitelné pro uložení dalších souborů)? Detailně vysvětlete proč.

Otázka č. 2 (X)

O souboru `x.au` víme, že obsahuje zvuková data uložená jako linear PCM (Pulse Code Modulation – uložená hodnota lineárně odpovídá přímo amplitudě vstupního signálu, a hodnoty jsou uložené jako znaménková čísla ve dvojkovém doplňku) – bližší specifikace `.AU` formátu **viz příloha**. Rozhodněte a zdůvodněte, jak dlouhý zvukový záznam (uveďte v milisekundách) je v souboru `x.au` uložený. Nakreslete přibližný tvar křivky uloženého zvukového záznamu.

Otázka č. 3 (X)

Předpokládejte, že spustíme níže uvedený program v Pascalu na počítači s běžným desktopovým procesorem Intel i7, a tím přepíšeme začátek uvedeného souboru `x.au` (zbytek souboru zůstane zachován). Pokud bychom poté obsah souboru opět zobrazili pomocí hex vieweru, tak napište, jaký bude obsah prvních 10 bytů takového souboru (typ *single* je 32-bit číslo dle IEEE 754, tj. mantisa je normalizována se skrytou 1 a zabírá spodních 23 bitů, pak následuje 8-bit exponent uložený ve formátu bias +127, poslední bit, tedy `MSb`, je znaménkový).

```
var
  s : single; bytesWritten : longint;
  f : file of byte;
begin
  Assign(f, 'x.au'); Reset(f);
  s := 579.75;
  BlockWrite(f, s, sizeof(s), bytesWritten);
  Close(f);
end.
```

Otázka č. 4 (X)

Jak bude vypadat Pascalový záznam (record), který by popisoval hlavičku `.AU` souboru? Napište v Pascalu program, který je schopen přečíst informace z libovolného souboru `x.au` ve formátu `.AU` **s 16-bit vzorky**, a který pro takový soubor zobrazí délku uložených zvukových dat v milisekundách. Bylo by možné v takovém programu použít uvedený záznam pro čtení a analýzu struktury souboru `x.au`?

Otázka č. 5

Předpokládejte následující deklarace (kde typ `longword` je 32-bitový celočíselný bezznaménkový, typ `word` je 16-bitový celočíselný bezznaménkový):

```
type
  PLongword = ^longword; PWord = ^word;
procedure Conv(src : PLongword; dst : PWord);
```

Napište ve Free Pascalu implementaci procedury `Conv` tak, aby převedla vstupní textový null-terminated řetězec `src` z kódování UTF-32 LE do kódování UTF-16 LE a výsledné znaky UTF-16 LE null-terminated řetězce uložila do paměti na místo, kam ukazuje argument `dst` (předpokládejte, že váš kód poběží pouze na little-endian platformách, a že na místě, kam ukazuje proměnná `dst`, je dostatek nevyužité paměti).

Unicode znaky z rozsahu `$010000` až `$10FFFF` se v UTF-16 kódují následujícím způsobem:

(1) Od kódu znaku se odečte hodnota `$010000`, a výsledné 20-bitové číslo se rozdělí na dvě 10-bitové části, které se zakódují dle následujících pravidel.

(2) Nejvyšších 10-bitů 20-bitové hodnoty se uloží do nejnižších 10 bitů prvního 16-bitového surrogate znaku (leží na nižší adrese). Horních 6 bitů první surrogate má být nastaveno na (vlevo je hodnota bitu 15, vpravo bitu 10): `1101 10`

(3) Nejnižších 10-bitů 20-bitové hodnoty se uloží do nejnižších 10 bitů druhého 16-bitového surrogate znaku (leží na vyšší adrese). Horních 6 bitů druhé surrogate má být nastaveno na (od první surrogate se liší pouze 10. bitem): `1101 11`

Otázka č. 6

Detailně vysvětlete, co to je tzv. *proces*, a v kontextu nějakého moderního OS (jako Linux nebo Windows) napište, jaké hlavní celky budou součástí *kontextu procesu* (ke každé napište stručné vysvětlení, o co se jedná).

Společná část pro otázky označené Y

Předpokládejte, že máme k dispozici ADC (analogově-digitální převodník) typu `ADC121C027` od společnosti Texas Instruments. S převodníkem se komunikuje pomocí standardní varianty sběrnice I²C. Datasheet tohoto ADC najdete **v příloze**.

Otázka č. 7 (Y)

Předpokládejte, že navrhujeme záznamovou zvukovou kartu připojitelnou na standardní systémovou sběrnici PCI – tj. naše zvuková karta nebude podporovat přehrávání zvuků, ale **pouze nahrávání** ze **stereo** analogového zvukového zdroje do počítače. Ve zvukové kartě budeme analogová data převádět do digitální podoby pomocí dvojice výše uvedených ADC převodníků `ADC121C027`. Navrhněte, jak by mohlo vypadat HCI takové zvukové karty – tj. napište jaké registry bude muset taková zvuková karta minimálně obsahovat, a pro každý takový registr popište jeho funkci a obsah (a uveďte zda je pro čtení, či pro zápis). Předpokládejte, že zvuková karta má podporovat pouze PIO přenosy.

Otázka č. 8 (Y)

V příloze najdete datasheet RS-232 to I²C bridge NXP Semiconductors `SC18IM700`, který **známe z přednášky**. Na jeho I²C sběrnici připojíme jeden ADC převodník `ADC121C027` (na pinu `ADDR` je připojené kladné napájecí napětí). Z tohoto ADC převodníku bychom nyní chtěli přečíst právě naměřenou hodnotu připojeného analogového signálu (předpokládejte, že převodník naměřil hodnotu, jejíž digitální reprezentací je hodnota 280_{10}, kterou nám převodník vrátí). Napište **v šestnáctkové** soustavě hodnotu všech bytů, které musíme po RS-232 lince poslat RS-232 to I²C bridge tak, aby se provedlo přečtení právě jedné naměřené hodnoty z připojeného ADC převodníku. Dále **v šestnáctkové** soustavě napište hodnotu všech bytů, které nám po RS-232 lince dorazí od RS-232 to I²C bridge jako odpověď.

Otázka č. 9

Předpokládejte níže popsaný CPU vycházející z architektury procesorů Intel 80386 – je to **32-bitový little-endian** CPU s obecnou registrovou architekturou, s podporou stránkování a s 32-bitovým virtuálním i fyzickým adresovým prostorem. Procesor má obecné registry `EAX`, `EBX`, `ECX`, `EDX`, `DSI`, `EDI`, `EBP`, 32-bitový příznakový registr `EFLAGS` s běžnými příznaky, registr `ESP` (stack pointer, ukazuje na poslední využitý byte, roste dolů), a registr `EIP` (instruction pointer). V instrukční sadě jsou mimo jiné **i následující instrukce** (příznakový registr modifikují pouze aritmetické operace, ale instrukce přenosu dat nikoliv):

- `MOV reg,DWORD PTR imm32/[addr]` (load register)
- `MOV DWORD PTR [addr],reg` (store register)
- `MOV DWORD PTR [addr],imm32` (store constant)
- `MOV reg0,reg1` (transfer from `reg1` to `reg0`)
- `ADD reg,imm32/[addr]/reg` (add without carry)
- `SUB reg,imm32/[addr]/reg` (subtract without carry)
- `PUSH imm16/imm32/[addr]/reg`, `POP [addr]/reg`
- `JMP addr` (direct jump), `JNE addr` (jump if not equal)
- `CMP DWORD PTR [addr],imm32` (32-bit compare)
- `CALL addr` (direct call)
- `RET` (return from subroutine)

Všechny výše uvedené instrukce se dvěma operandy mají vždy **vlevo cílový** a **vpravo zdrojový** operand. Instrukce mohu mít jednu z následujících variant operandů (povolené varianty viz definice konkrétní instrukce):

- 32-bitový immediate `imm32`
- absolutní adresa `[addr]`, kde `[addr]` může být:
  - `[reg +/- imm]` adresa daná součtem/rozdílem obsahu registru `reg` a konstanty `imm`
- libovolný registr `reg`

Napište v Pascalu bez použití inline assembleru kód funkce (i s deklarací), která by mohla být běžným překladačem přeložena do níže uvedeného kódu v assembleru 80386 (předpokládejte, že funkce používá běžnou Cčkovou volací konvenci, tj. argumenty se předávají na volacím zásobníku zprava doleva, a **odebírá je volající**, návratová hodnota je uložená v registru `EAX`):

```
LABEL1:
  push  ebp
  mov   ebp,esp
  sub   esp,0x8
  cmp   DWORD PTR [ebp+0x8],0x0
  jne   LABEL2
  mov   DWORD PTR [ebp-0x4],0x0
  jmp   LABEL4
LABEL2:
  cmp   DWORD PTR [ebp+0x8],0x1
  jne   LABEL3
  mov   DWORD PTR [ebp-0x4],0x1
  jmp   LABEL4
LABEL3:
  mov   eax,DWORD PTR [ebp+0x8]
  sub   eax,0x1
  push  eax
  call  LABEL1
  add   esp,0x4
  mov   DWORD PTR [ebp-0x8],eax
  mov   eax,DWORD PTR [ebp+0x8]
  sub   eax,0x2
  push  eax
  call  LABEL1
  add   esp,0x4
  add   eax,DWORD PTR [ebp-0x8]
  mov   DWORD PTR [ebp-0x4],eax
LABEL4:
  mov   eax,DWORD PTR [ebp-0x4]
  mov   esp,ebp
  pop   ebp
  ret
```

Otázka č. 10

Vysvětlete, co to je tzv. *object file*, a popište z jakých hlavních částí se typický formát *object* souborů skládá. V tomto kontextu vysvětlete, co to je tzv. *statické linkování*, kdo a jak ho provádí, a k čemu se využívá.