Questão 1 (valor 2,0 pontos)

Simule a execução do programa abaixo, destacando a sua saída. A saída do programa consiste de tudo que resulta dos comandos `printf`.

```
# include <stdio.h>

int f1 (int a, int b) {
  int z;

  a = a + b;
  b = a - b;
  z = a + b;

  return z;
}

int f2 (int *a, int b) {
  int z;

  *a = *a + b;
   b = *a - b;
   z = *a + b;

  return z;
}

int f3 (int *a, int b) {

  b = *a + b;
  *a = b + 2;

  return b;
}
```

```
int main () {
  int nusp;
  int a, b, c, d, e;
  double f;

  printf ("Entre com seu no. USP: ");
  scanf ("%d", &nusp); /* use aqui seu numero USP */
  printf ("nusp = %d\n", nusp);

  a = nusp % 5;
  b = 6 - a;

  printf ("1: a=%d b=%d\n", a, b);

  c = a;
  d = b;
  e = f1 (c, d);

  printf ("2: a=%d b=%d c=%d d=%d e=%d\n", a, b, c, d, e)

  c = a;
  d = b;
  e = f2 ( &c, d);

  printf ("3: a=%d b=%d c=%d d=%d e=%d\n", a, b, c, d, e)

  d = b;
  d = f3 ( &d, d);

  printf ("4: a=%d b=%d d=%d\n", a, b, d);

  d = 2 * a + 1;
  b = 2;
  f = d / b;

  printf ("5: a=%d b=%d d=%d f=%f\n", a, b, d, f);

  f =  a;
  f = (2 * f + 1) / 2;
  e = f;

  printf ("6: a=%d b=%d e=%d f=%f\n", a, b, e, f);

  return 0;
```

Questão 2 (valor 4,0 pontos)

Frequentemente, quando somos solicitados a criar uma senha, devemos fornecer uma senha que não seja fácil de ser "quebrada". Assim, algumas recomendações nos são fornecidas, tais como senhas que devam ter pelo menos um certo números de caracteres e uma mistura de letras maiúsculas e minúsculas; dígitos e símbolos.

PARTE A. (valor 2,0 pontos) Escreva uma função em C com protótipo

```
char conta_char (int *n_upper, int *n_lower, int *n_digs, int *n_others);
```

que lê uma sequência de caracteres terminada por um espaço (em branco), que para fins de visualização é representado aqui por '␣'; ou por um ponto '.' (ou seja, esses caracteres não pertencem à sequência) e devolve a quantidade de letras maiúsculas, letras minúsculas, dígitos e outros caracteres em `*n_upper`, `*n_lower`, `*ndigs` e `*n_others`, respectivamente. A função devolve via `return` o caractere que finaliza a sequência, ou seja, ou '␣' (espaço) ou '.' (ponto).

Exemplos: para as sequências

Sequência	`*n_upper`	`*n_lower`	`*ndigs`	`*n_others`	`return`
"As1001noites␣"	1	7	4	0	'␣'
"50naoeh20."	0	5	4	0	'.'
"20BuScar,maS100Pressa␣"	4	11	5	1	'␣'

PARTE B. (valor 2,0 pontos)

Para fins desta questão, uma senha é *válida* se, e somente se, ela contém pelo menos uma letra maiúscula, pelo menos uma letra minúscula, pelo menos um número, e pelo menos 8 caracteres. Adicionalmente, uma senha válida deve conter somente números e letras, ou seja, não deve conter símbolos.

Exemplos:

- A sequência "As1001noites" é uma senha válida.
- As sequências "50naoeh20" e "20BuScar,maS100Pressa" não são senhas válidas.

Escreva um programa em C que lê uma sequência não vazia de caracteres (ou seja, ela tem pelo menos um caractere que seja uma letra, um dígito ou um símbolo), seguida por um ponto '.' (isto é, o ponto não pertence à sequência), e imprime qual é a proporção de senhas válidas.

Exemplos:

Sequência	Impressão
"20Buscar␣80cAo␣80Cao20Buscar␣SIMPLEs."	0.50
"As1001noites␣50naoeh20␣20BuScar,maS100Pressa."	0.33

Observação: Use, obrigatoriamente, a função **conta_char** da **PARTE A**. Não é necessário reescrever a função aqui. Use a função mesmo que você não tenha feito a **PARTE A** da questão.

Questão 3 (valor 4,0 pontos)
PARTE A. (valor 2,0 pontos) Escreva uma função em C com protótipo

```
double arctan (double x, double eps);
```

que recebe dois valores reais x e eps, com $|x| \leq 1.0$, e $0.0 < \text{eps} < 1.0$; e devolve o arco tangente de x com precisão eps.

Para calcular o arco tangente de x com uma certa precisão eps, utilize a fórmula abaixo e some os termos até que um termo da série tenha valor absoluto menor que eps. O primeiro termo com valor absoluto menor que eps deve ser somado.

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots + (-1)^k \frac{x^{2k+1}}{2k+1} + \cdots$$

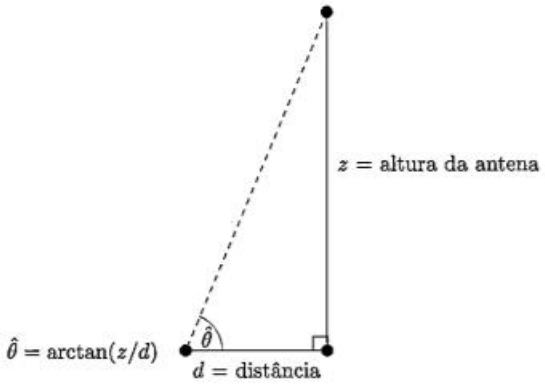
Observações:

- Não é permitido usar as funções da biblioteca `<math.h>`.
- Você pode declarar a sua própria função para calcular o valor absoluto de um número real.

PARTE B. (valor 2,0 pontos) Desconfiados da qualidade dos dispositivos secretos instalados nos carros dos politeranos, os calouros de 2018 decidiram implementar um sistema de aferição das medições providas pelos sensores. Escreva um programa em C que lê (via teclado mesmo):

- o número n de casos de teste;
- para cada caso de teste, lê:
  - a altura z de uma antena;
  - a distância d (calculada pelo EP2, da base desta antena ao carro dos politeranos);
  - o ângulo de incidência θ (fornecido pelo dispositivo secreto instalado no carro).

Para cada caso de teste, o seu programa deve calcular o ângulo de incidência estimado $\hat{\theta}$ a partir de z e d, como $\hat{\theta} = \arctan(z/d)$, conforme a figura abaixo, e ao final deve imprimir o número de casos de teste em que o ângulo calculado difere de mais do que `delta` do respectivo ângulo θ (ou seja, $|\theta - \hat{\theta}| >$ `delta`). Além disso, se o número de casos com divergência for maior do que 10% do total de casos de teste (ou seja, mais que $0.1 * n$), o programa deve imprimir também a mensagem de alerta: "*Equipamento com Problema*".



Seu programa deve:

- Usar `delta` = 0.01.
- Usar, obrigatoriamente, a função **arctan** da **PARTE A**, com valor eps=0.001. Não é necessário reescrever a função aqui. Use a função mesmo que você não tenha feito a **PARTE A** da questão.

Observações:

- Não é permitido usar as funções da biblioteca `<math.h>`.
- Você pode declarar a sua própria função para calcular o valor absoluto de um número real.