MAC2166 Introdução à Computação para Engenharia

Escola Politécnica

Primeira Prova — 07 de abril de 2008

Nome: _____

Assinatura: _____

Nº USP: _____ Turma: _____

Professor: _____

Instruções:

1. Não destaque as folhas deste caderno.
2. A prova consta de 4 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
3. A prova pode ser feita a lápis. Cuidado com a legibilidade.
4. Qualquer questão pode ser resolvida em qualquer página. Se a questão não estiver na página correspondente ao enunciado basta indicar isto na página e escrever QUESTÃO i em letras ENORMES antes da solução.
5. Não é necessário apagar rascunhos no caderno de questões.
6. Não é permitido o uso de folhas avulsas para rascunho.
7. Não é permitido o uso de calculadoras.
8. Não é permitido a consulta a livros, apontamentos ou colegas.

DURAÇÃO DA PROVA: 2 horas

Questão	Valor	Nota
1	2,5	
2	2,5	
3	2,5	
4	2,5	
Total	10,0	

Questão 1 (valor: 2,5 pontos)

Simule a execução do programa abaixo, destacando a saída do programa. A saída do programa é tudo que resulta dos comandos printf. Para efeito de correção só será considerada a saída do programa.

```
#include <stdio.h>

int main()
{
  int x, y, cont, z;

  printf("Digite o digito final do seu Numero USP: ");
  scanf("%d", &x);

  printf("Primeiro: x = %d \n", x);

  y = 10 * (2 + 3 * (x%2));
  printf("Segundo: y = %d \n", y);

  z = 15;

  if (z > x && y < 40)
  {
     y = y + 5;
     printf("Bom: y = %d\n", y);
  }
  else
  {
     y = y - 25;
     printf("Otimo: y = %d\n", y);
  }

  cont = 1; y = 10;

  while (cont < 3 && (y <= 5 || z > 3))
  {
      if (x%2 == 0)
      {
          y = y - 2;
          z = z + 1;
          printf("Otimo: y = %d z = %d\n", y, z);
      }
      else
      {
          z = z - 1;
          y = y + 3;
          printf("Bom: y = %d z = %d\n", y, z);
      }

      x = x + 3;
      printf("cont = %d x = %d\n", cont, x);
      cont = cont + 1;
  }

  printf("Ultimo: cont = %d\n", cont);

  return 0;
}
```

Para efeito de correção só será considerada a saída do programa. Você pode usar a tabela abaixo como bem entender. Cada turma está habituada a simular de maneira diferente, fazendo tabelas com "caras" diferentes.

saída

Questão 2 (valor: 2,5 pontos)

Dados dois inteiros a e b, com $a \geq 0$ e $b \geq 1$, os seguintes trechos de programa têm como objetivo calcular o quociente e o resto da divisão de a por b. No final, a variável q deve ter valor do quociente e a variável r deve ter o valor do resto. Indique se o trecho está correto ou incorreto. Para cada trecho indicado como incorreto, dê um valor de a e b para o qual a resposta é incorreta.

(a) Correto ( ) Incorreto ( )

```
q = 0;
r = a;
while ( r >= b ) {
  r = r - b;
  q = q + 1;
}
```

(b) Correto ( ) Incorreto ( )

```
q = a / b;
r = a % b;
```

(c) Correto ( ) Incorreto ( )

```
for( q=0; (q+1)*b <= a; q=q+1 ) {
  r = q;
}
r = a - q * b;
```

(d) Correto ( ) Incorreto ( )

```
q = 0;
while ( (q+1)*b <= a ) {
  r = a - q * b;
  q = q + 1;
}
```

(e) Correto ( ) Incorreto ( )

```
q = 1;
r = a - b;
while ( r >= b ) {
  r = r - b;
  q = q + 1;
}
```

Questão 3 (valor: 2,5 pontos)

Escreva um programa em C que leia um número inteiro $n > 0$ e calcule quantos pares de dígitos consecutivos de mesma paridade esse número contém. Por exemplo, o número 1324 contém dois pares de dígitos consecutivos de mesma paridade (o par 13 e o par 24), assim como o número 973 também contém 2 pares de dígitos consecutivos de mesma paridade (o par 97 e o par 73). Já o número 12345, não contém nenhum par.

Exemplos de seis possíveis execuções do programa:

Exemplo 1:

```
Digite um numero n>0:  2
Resposta:  nenhum par
```

Exemplo 3:

```
Digite um numero n>0:  281
Resposta:  1 par(es) de digitos
```

Exemplo 5:

```
Digite um numero n>0:  12370
Resposta:  1 par(es) de digitos
```

Exemplo 2:

```
Digite um numero n>0:  23
Resposta:  nenhum par
```

Exemplo 4:

```
Digite um numero n>0:  1234
Resposta:  nenhum par
```

Exemplo 6:

```
Digite um numero n>0:  24003
Resposta:  3 par(es) de digitos
```

Questão 4 (valor: 2,5 pontos)

No *GrayJack*, dada a pontuação do apostador e o número de cartas com o qual o apostador obteve essa pontuação, a banca deve se dar cartas tentando vencer o apostador. Sabemos que o banca vencerá o apostador se não estourar (pontuação maior do que 21) e obtiver uma pontuação maior que a do apostador ou, no caso de ter a mesma pontuação do apostador, ter um número menor ou igual de cartas.

Escreva um programa em C que leia do teclado a pontuação do apostador (um inteiro maior do que zero) e o número de cartas com o qual o apostador obteve essa pontuação (um inteiro maior do que zero). O programa deve simular o jogo da banca e indicar quem ganhou.

Observações:

1. Lembre-se de que para sortear uma carta você deve

   (a) colocar o `#include <stdlib.h>`,

   (b) e usar o comando:

   ```
   carta = 1 + (int) (13.0 * (rand() / (RAND_MAX + 1.0)));
   ```

2. Com relação a este último comando, para que ele funcione corretamente é necessário colocar no início do programa (na primeira linha depois da declaração da última variável, e somente aí) a *inicialização da semente*. Para isso, use o comando:

   ```
   srand (321);
   ```

3. Seu programa deve verificar quem ganha o jogo independemente da situação do apostador ser possível ou não. Por exemplo: pontuação do apostador igual a 2 (dois) com o número de cartas igual a 10 (dez). Em outras palavras, seu programa não precisa verificar se a situação do apostador é possível ou não!