	Universidade Federal de Campina Grande **Centro de Engenharia Elétrica e Informática** **Departamento de Sistemas e Computação** **Disciplina: *Técnicas de Programação*** **Prof.: José Eustáquio Rangel de Queiroz** **EXERCÍCIO DE SONDAGEM – TURMA 01** **SEMESTRE 2016-2 DATA: 01/11/2016**

Professor de INTRODUÇÃO à PROGRAMAÇÃO		
Matrícula	**Nome**	**Nota**

Obs.: As questões deste exercício deverão ser solucionadas conforme seus enunciados. As questões de 01 a 27 deverão ser feitas na primeira metade da aula. As questões de codificação deverão ser realizadas em DUPLAS, na segunda metade da aula. Para as questões de implementação de código (28 a 30), deverão ser gerados os códigos-fontes e os executáveis (trocar a extensão EXE por DOT).

1. Por ser uma linguagem fortemente tipada, a sintaxe das declarações de variáveis em C necessita obrigatoriamente a inclusão do nome de um tipo ao nome da variável declarada.

A.	Verdadeiro
B.	Falso

2. Em se tratando de operações de atribuição, se o tipo do lado esquerdo da igualdade não for o mesmo do lado direito, o tipo do lado direito será convertido para o tipo do lado esquerdo, sem que tal conversão implique perda de precisão em nenhum tipo de dado permitido pela linguagem C.

A.	Verdadeiro
B.	Falso

3. Assinale a opção que inclui somente nomes *inválidos* para variáveis na linguagem C.

A.	***If, ab237, H789, &ya***
B.	***X_ou_Y, j3, Int, obs****
C.	***y_ou_z, a36, includ, stdIO***
D.	***2_ou_1, \fim, *h|j***
E.	Nenhuma das opções anteriores.

4. Em C, constantes de caractere são representadas entre *aspas* (“”) e equivalem ao número pelo qual o caractere é representado na máquina.

A.	Verdadeiro
B.	Falso

5. O programa a seguir:

```
#include <stdio.h>

int main(){
  int *a, b;
  scanf("%d", &a, b);
  if (&a==b) printf("%d eh igual a %d", a, b);
```

```
}
```

lê duas variáveis via teclado, imprimindo que uma delas é igual à outra após um teste de igualdade.

A.	Verdadeiro
B.	Falso

6. Considerando-se o trecho de código a seguir, escrito em linguagem C, no qual x e y são variáveis do tipo int e a é um apontador para uma variável do tipo int:

```
a = &x;
y = *a;
```

É correto afirmar que:

A.	o valor de ***a*** será atribuído a ***y***.
B.	o endereço de ***x*** será atribuído a ***y***.
C.	o endereço de ***a*** será atribuído a ***y***.
D.	o valor de ***x*** será atribuído a ***y***.
E.	Nenhuma das alternativas anteriores está correta.

7. Após a execução do seguinte programa escrito em C:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
     int x, y, z, tab[4][3];

     for (x=0; x<4; ++x)
          for (y=0; y<3; ++y)
              tab[x][y]=(x*4)+y+1;
              z= 3*tab[2][2] - 2*tab[1][1];
         printf("%d\n%d\n", z, tab[x-2][1]*tab[1][2-y]);
}
```

é correto afirmar que:

a.	***tab[3][2]*** receberá o valor ***15***.
b.	Um dos valores impressos será ***30***.
c.	***z*** receberá o valor ***21***.
d.	***12*** posições do *array* bidimensional ***tab*** serão preenchidas.
e.	Todas as alternativas anteriores estão corretas.

8. A saída do trecho de código a seguir, escrita em linguagem C,

```
#include <stdio.h>

void f1(int *a, int *b)
{
  int k;
  k = *a;
  *a = *b;
  *b = k;
}
int main ()
{
  int x = 8, y = 6;
  f1 (&x,&y);
  printf ("%i %i\n",x,y);
}
```

será:

a.	***8 6***
b.	***6 8***
c.	***8*** ***6***
d.	***6*** ***8***
e.	Todas as alternativas anteriores estão incorretas.

9. O trecho de código a seguir, escrito em linguagem C:

```
#include <stdio.h>

long int func(int n)
{
     if (n==0)
     return 1;
     else
     return n*func(n-1);
}

int main ()
{
     printf ("%i\n", func(10)%11);
 }
```

a.	Utiliza passagem de parâmetros por referência.
b.	Gera como saída o valor **11**.
c.	Produz o fatorial de um número.
d.	Utiliza recursos iterativos.
e.	Nenhuma das alternativas anteriores está correta.

10. O seguinte programa em C:

```
#include <stdio.h>
int main(){
  int i =2;
  printf ("\n O valor de i = %d ", i);
}
```

a.	Não executa nada.
b.	Imprime: O valor de ***i*** = ***2***.
c.	Imprime: ***\n O valor de i = %d***.
d.	Salta para a próxima linha e imprime: ***O valor de i = 2***.
e.	Todas as alternativas anteriores estão corretas.

11. Sobre o trecho de código a seguir, salvo em seu computador na pasta ***C:\temp\TP*** com o nome ***teste***:

```
#include <stdio.h>
#include <stdlib.h>

int main ( int argc, char * argv[] )
{
    int i, s;
    s = 0;
    for (i = 1; i <= argc ; i++)
    s = s + atoi(argv [i]);
    printf ("Soma dos numeros: %d\n",s);
}
```

pode-se afirmar que:

a.	***argc*** é um inteiro que conterá o número de argumentos com os quais o programa for chamado na linha de comando.
b.	***argv*** é um apontador para um vetor de *strings*, cada uma das quais corresponderá a cada um dos parâmetros digitados quando o programa for chamado na linha de comando.
c.	Embora os nomes ***argc*** e ***argv*** possam ser alterados, costuma-se não modificá-los, por questão de padronização.
d.	Produzirá como saída ***Soma dos números: 36*** se o comando ***C:\temp\TP\teste 3 7 11 15*** for digitado numa linha de comando.
e.	Todas as alternativas anteriores estão corretas.

12. O comando ***printf ("%s%d%","Taxa de ",1.3);*** produzirá a impressão:

a.	***Taxa de 1.3%***
b.	***%s%d%% Taxa de 1.3***
c.	***%Taxa de 1.3%***
d.	***%Taxa de 1.3***
e.	Nenhuma das alternativas anteriores.

13. Acerca da linguagem C, pode-se afirmar que:

a.	Se uma função não retorna nada, ela deverá ser declarada como ***void***.
b.	O retorno da função ***main()*** é feito para o sistema operacional.
c.	***stdio.h*** e ***string.h*** contêm protótipos de algumas funções da biblioteca do C.
d.	Funções podem ser definidas dentro de outras funções.
e.	Uma das opções anteriores é falsa.

14. A saída do trecho de código a seguir:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf ("O resultado eh: %d\n",(54 || 0)||(300 >= 28)&&(1 != 0)&&(1 < 0));
}
```

é:

a.	***O resultado eh: 0***.
b.	***O resultado eh: 1***.
c.	***O resultado eh: 2***.
d.	Nada se pode afirmar.
e.	Nenhuma das opções anteriores.

15. ***if(!num) ...;*** é equivalente a ***if(num!=0) ...;***

a.	Verdadeiro
b.	Falso

16. Os trechos de programa a seguir

```
for (j = 10 ; j > 0; j--) printf("%d\n", j);
```
e
```
for (j = 10 ; j > 0; --j) printf("%d\n", j);
```
são equivalentes entre si, sob o ponto de vista do que é impresso:

a.	Verdadeiro
b.	Falso

17. O trecho de programa a seguir

```
switch(!num)
{
  case 1:
   printf("O valor eh 10");
   break;
  case 2:
   printf("O valor eh 20");
   break;
  default:
   printf("O valor não eh 10 nem 20");
   break;
}
```
é:

a.	Válido na linguagem C.
b.	Inválido na linguagem C.

18. Os dois blocos de código a seguir

```
for(i = 0; i < 3; i++)
  for (j = 0; j < 3; j++)
    printf("i+j = %d \n", i+j);
```
e
```
for(i = 0, j=0; i < 3; i++)
  for ( ; j < 3; j++)
    printf("i+j = %d \n", i+j);
```
produzem o mesmo resultado.

a.	Verdadeiro
b.	Falso

19. Sendo ***i*** declarado e inicializado como ***int i = 100;***, os seguintes extratos de código:

```
while (i < 5){
  printf("%d %d %d \n", i, i-5, i-10);
  i = 5;
}
```
e
```
if (i <5) printf ("%d %d %d \n", i, i-5, i-10);
```

a.	São idênticos sob o ponto de vista da informação apresentada em tela.
b.	Não imprimem nenhuma informação na tela.
c.	Apresentam erros sintáticos.
d.	Um deles imprime ***100***, ***95*** e ***90*** uma única vez e o outro entra em laço infinito.
e.	Nenhuma das opções anteriores.

20. O trecho de código a seguir:

```
#include <stdio.h>
int main(void){
 printf ("O resultado da expressao eh: %x \n", ((255 & 150)&(199 & 234))<<2);
 return 0;
}
```

produzirá a saída:

a.	***O resultado da expressao eh: 0***.
b.	***O resultado da expressao eh: fe***.
c.	***O resultado da expressao eh: c***.
d.	***O resultado da expressao eh: 208***.
e.	Nenhuma das opções anteriores.

21. A saída produzida pelo extrato de código a seguir:

```
int x;
  for(x = 57; x > 1; x/=3) printf("%x " , x>>1);
```

será:

a.	***28 9 3 1***
b.	***34 11 3 1***
c.	***1c 9 3 1***
d.	***72 26 c 4***
e.	Nenhuma das opções anteriores.

22. Os extratos de código a seguir:

```
        int x = 15;
        while(--x > 7){
          printf("%d", x);
        }
```

e

```
        int x = 15;
        do{
          printf("%d", x-1);
        }while(--x > 8);
```

são equivalentes entre si.

a.	Verdadeiro
b.	Falso

23. A estrutura do ***switch*** a seguir:

```
switch (temp){
  case temp < 10:
    printf("Ceu nebuloso com possiveis ocorrencias de chuva");
    break;
  case temp < 25:
    printf("Ceu claro");
    break;
  default:
    printf("Ceu instavel");
}
```

é:

a.	Válida na linguagem C
b.	Inválida na linguagem C

24. Sobre o resultado do comando a seguir:

```
if((temp>19) ? printf("A cultura de fungos tem mais 75""%%"" de chances de sobreviver.")
: printf("A cultura de fungos tem menos de 5""%%"" de chances de sobreviver."))
```

se o valor de ***temp*** for igual a **19**, pode-se afirmar que:

a.	Será impressa a mensagem ***A cultura de fungos tem mais 75""%%" " de chances de sobreviver.***
b.	A sintaxe está incorreta.
c.	Será impressa a mensagem ***A cultura de fungos tem mais 75% de chances de sobreviver.***
d.	Nada será impresso.
e.	Nenhuma das opções anteriores.

25. Sobre o resultado da execução do programa a seguir:

```
#include <stdio.h>
#include <string.h>

int main(){
   struct Data{
      int Dia;
      char Mes[12];
      int Ano;
   };
   struct Data hoje, *apont=&hoje;
   hoje.Dia = 1;
   strcpy(hoje.Mes, "Novembro");
   hoje.Ano = 2016;
   printf("A data de hoje eh %d de %s de %d\n", hoje.Dia, hoje.Mes, apont->Ano);
   return 0;
}
```

pode-se afirmar que:

a.	Nada será impresso, porque o programa contém erros de sintaxe.
b.	Será impressa a mensagem: ***A data de hoje eh 01 de Novembro de 2016***.
c.	Será impressa a mensagem: ***A data de hoje eh 01 de Novembro de 2293596***.
d.	Todas as opções acima estão incorretas.

26. Analise o programa a seguir:

```
#include <stdio.h>
#include <string.h>

struct Ponto{
  int x,y; _____________________________________________________________
};
void compara(struct ponto p1, struct ponto p2){
  char h[10],v[10]; ___________________________________________________
   ____________________________________________________________________
```

```
  if (p1.x > p2.x) {
    strcpy(h,"direita"); _____
  }
 else {
    strcpy(h,"esquerda"); _____
  }

  _____
    strcpy(v,"abaixo"); _____
 else {
    strcpy(v,"acima"); _____
  }

  printf("O ponto p1 estah a %s e %s do ponto p2\n",h,v);
}

int main() {
  struct ponto ponto1={2,5}, ponto2={7,9}; _____
  compara(ponto1,ponto2); _____
return 0;
}
```

Preencha as linhas de comentários e especifique a saída do programa.

27. Escrever um programa em C que chame duas funções, assim discriminadas:
    a) A primeira deve receber do usuário uma quantidade de tempo em segundos e convertê-la em horas, minutos e segundos, armazenando-a em seguida em uma variável ***horario***;
    b) A segunda deve ler a variável ***horario*** e devolver a quantidade de minutos em segundos.

28. Seja uma estrutura para descrever os carros de uma determinada revendedora, contendo os seguintes campos:

    marca: string de tamanho 15
    ano: inteiro
    cor: string de tamanho 10
    preco: real

    a. Escrever a definição da estrutura ***Carro***.
    b. Declarar o vetor ***vetcarros*** do tipo da estrutura definida acima, de tamanho **20** e global.
    c. Definir uma função para ler o vetor ***vetcarros***.
    d. Definir uma função que receba um preço e imprima os carros (***marca***, ***cor*** e ***ano***) que tenham preço igual ou menor ao preço recebido.
    e. Definir uma função que leia a marca de um carro e imprima as informações de todos os carros dessa marca (***preco***, ***ano*** e ***cor***).
    f. Definir uma função que leia uma marca, ano e cor e informe se existe ou não um carro com essas características. Se existir, informar o preço.

29. Seja um texto contendo vários nomes armazenados em uma matriz ***45 x 60***, na qual cada linha contém um nome completo de uma pessoa. As palavras de cada nome são separadas por espaços em branco, sendo o último sobrenome seguido por um ***\0***. Escrever um programa contendo as seguintes funções:
    a) ***le()*** - ler todos os nomes completos, armazenando-os na matriz.
    b) ***ordena_matriz()*** - ordenar a matriz de nomes completos.
    c) ***imprime_maiornome()*** - imprimir o maior nome, caso tenha empate, deve

imprimir todos os maiores nomes.

d) ***armazena_primnome()*** - percorrer a matriz de nomes e armazenar o primeiro nome de cada pessoa em um vetor de estruturas, bem como o índice da linha da matriz onde este apareceu e o número de ocorrências do nome recebe ***1***.
OBS.: Caso o primeiro nome já tenha aparecido antes (e.g., *José Augusto Silva* e *José Augusto Sousa*), não deve ser armazenado novamente, porém a linha da matriz onde este aparece deve ser armazenada e o número de ocorrências deve ser incrementado. Supor que cada nome aparece no máximo 10 vezes.

e) ***proc_nome()*** - le um primeiro nome (e.g., José) e imprime o nome completo de todas as pessoas que iniciam com esse nome (e.g., José da Silva, José de Alencar, José Augusto dos Anjos, etc.).

f) ***maior_ocorr()*** - imprime o primeiro nome que mais apareceu no texto, se houver empate, deve imprimir todos.

Obs.:

- A definição da estrutura ***Prim_Nome*** é:

```
struct Prim_Nome{
   char pnome[11]; // primeiro nome
   int linhas[10]; // linhas onde ocorre primeiro nome
   int n_ocor; // número de ocorrências do primeiro nome
 };
```

- A declaração do vetor de primeiros nomes é: ***struct Prim_Nome vet_nome[30]***;
- Utilize as funções para manipulação de strings definidas em string.h.

30. Escrever um programa em C que apresente em tela uma sequência de ***30*** números que obedeçam à seguinte regra: (i) os dois primeiros números são ***1***; e (ii) qualquer outro número corresponde à soma dos dois anteriores (***1 1 2 3 5 8 13 21 ...***). Esta seqüência é conhecida como *série de Fibonacci*. Usar funções.

Benvindos a Técnicas de Programação!