Universidade Federal Fluminense
Campus de Rio das Ostras
Instituto de Ciência e Tecnologia

Programação de Computadores II
Segunda Prova (P2) 28/11/2019
Prof. Alessandro Copetti

Nome: _____**GABARITO**_____

Obs.: A prova pode ser feita a lápis! As funções não devem conter valores fixos!

1. O que será impresso na tela? Mostre o valor que cada variável assume durante a execução e também circule a resposta; não é para explicar o que cada linha faz.

a) (1,5pt)

```
typedef struct {
    int x,y;
} Ponto;

void funcao(Ponto *k) {
    k->x = k->x + 2;
}

void funcao2(Ponto *v, int n) {
    int i=0;
    for (;i<n;i++) {
        v[i].x=v[i].x+2;
        v[i].y=v[i].y+2;
    }
}


main() {
   Ponto q[3]={{1,2},{2,4},{5,6}};
   Ponto *p = (Ponto *) malloc(sizeof(Ponto));
   p->x = 2;   p->y = 4;
   funcao(p);
   printf("\n%d",p->x+p->y);

   printf("\n%d",q[2].x+q[2].y);

   funcao2(q,3);
   printf("\n%d",q[2].x+q[2].y);

}
```

8
11
15

b) (2,5pts) Desenvolva uma função para guardar o vetor de estruturas em um arquivo. Você pode escolher se guardará em um arquivo texto (x,y em cada linha) ou em um arquivo binário. A função deve receber o número de elementos do vetor de pontos. Passe como parâmetro também o nome do arquivo para a função. Escreva a linha de código que será usada para chamar a função na última linha da main().

```
void gravaVetorPontos(Ponto *p, int n, char nomearq[50]) {
   FILE *fp = fopen(nomearq, "w");
   int i=0;
   for (;i<n;i++) {
      fprintf(fp, "%d,%d\n",p[i].x,p[i].y);
   }
   fclose(fp);
}
```

```
// Chamada da função:
//   gravaVetorPontos(q,3,"vetPontos.txt");
```

Com arquivos binários:

```
void gravaVetorPontos_binario(Ponto *p, int n, char nomearq[50]) {
   FILE *fp = fopen(nomearq, "wb");
   fwrite(p, sizeof(Ponto),n,fp);
   fclose(fp);
}

void leVetorPontos_binario(Ponto *p, int n, char nomearq[50]) {
   FILE *fp = fopen(nomearq, "rb");
   int i=0;
   fread(p, sizeof(Ponto),n,fp);
   for (;i<n;i++)
      printf("\nx=%d, y=%d",p[i].x,p[i].y);
   fclose(fp);
}

// Chamada da função:
//   gravaVetorPontos(q,3,"vetPontos.bin");
```

2. Tendo o seguinte código:

```
typedef struct {
    int dia, mes, ano;
} Data;

typedef struct {
    int matricula;
    char nome[80];
    Data admissao;
} Funcionario;

int main () {
    char nome[80];
    int matricula,d,m,a;
    int n, i;
    Lista *lista=NULL;
    Funcionario **f;

    printf("Informe a quantidade de funcionários: ");
    scanf("%d", &n);


    ____________________________________________________________

    for (i=0; i<n; i++) {
        printf ("digite o nome do funcionário ");
        scanf ("%79[^\n]",nome);
        printf("digite a matrícula do funcionário ");
        scanf ("%d", &matricula);
        printf("digite a data de admissão do funcionário ");
        scanf ("%d %d %d",&d,&m,&a);
        f[i] = criaFuncionario(matricula, nome, d, m, a);
        lista = insereLista(lista, matricula);
    }
    imprimeFuncionario(f,n);
}
```

a. (1pts) Escreva a linha de código (antes do 'for' da main) necessária para alocar o vetor de ponteiros de acordo com o número de funcionários informado pelo usuário.

b. (2pts) Desenvolva a função criaFuncionario que retorna uma nova estrutura funcionário.

c. (1pts) Desenvolva a função imprimeFuncionario que imprime todos os campos de todos os funcionários.

d. (2pts) Desenvolva a função insereLista, a qual insere um novo nó em uma lista, incluindo somente a informação de matrícula. Abaixo, tem a definição da Lista e uma função imprimeLista somente para consulta.

```
typedef struct lista Lista;
struct lista {
   Funcionario f;
   Lista *prox;
};

void imprimeLista(Lista *l) {
    Lista *p = l;
    printf("\nLista:");
    for (;p!=NULL;p=p->prox) {
        printf("\nMatricula:%d",p->f.matricula);
    }
}
```

```
typedef struct
{
   int dia, mes, ano;
} Data;

typedef struct
{
   int matricula;
   char nome[80];
   Data admissao;
} Funcionario;


Funcionario* criaFuncionario(int matricula, char* n, int d, int m, int a)
{
   Funcionario* funcionario = (Funcionario*)malloc(sizeof(Funcionario));
   strcpy(funcionario->nome,n);
   funcionario->matricula=matricula;
   funcionario->admissao.dia=d;
   funcionario->admissao.mes=m;
   funcionario->admissao.ano=a;
   return funcionario;
}


void imprimeFuncionario(Funcionario** vet, int n)
{
   int i;
   for(i=0; i < n; i++)
      printf("\nMatrícula=%d; Nome=%s; Dia=%d; Mês=%d; Ano=%d", vet[i]->matricula, vet[i]->nome, vet[i]->admissao.dia, vet[i]->admissao.mes, vet[i]->admissao.ano);
}

Lista *insereLista(Lista *l, int matricula) {
   Lista *novo = (Lista *) malloc(sizeof(Lista));
   novo->f.matricula = matricula;
   novo->prox = l;
   return novo;
}
```

```
void imprimeLista(Lista *l) {
   Lista *p = l;
   printf("\nLista:");
   for (;p!=NULL;p=p->prox) {
      printf("\nMatricula:%d",p->f.matricula);
   }
}

int main ()
{
   char nome [51];
   int matricula;
   int d,m,a;
   int n, i;
   Funcionario **f;

   printf("Informe a quantidade de funcionários: ");
   scanf("%d", &n);
   fflush(stdin);
   f = (Funcionario **) malloc(n * sizeof(Funcionario *));

   for (i=0; i<n; i++)
   {
      printf ("digite o nome do funcionário ");
      scanf ("%79[^\n]",nome);
      fflush(stdin);
      printf("digite a matrícula do funcionário ");
      scanf ("%d", &matricula);
      fflush(stdin);
      printf("digite a data de admissão do funcionário ");
      scanf ("%d %d %d",&d,&m,&a);
      fflush(stdin);
      f[i] = criaFuncionario(matricula, nome, d, m, a);
   }
   imprimeFuncionario(f,n);
}
```