INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA – CAMPUS CAMPINA GRANDE			
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PARAÍBA Campus Campina Grande	CURSO:	CURSO ENGENHARIA DA COMPUTAÇÃO	
	PERÍODO:		TURMA:
	DISCIPLINA:	PROGRAMAÇÃO E ESTRUTURAS DE DADOS	
	PROFESSOR:	CÉSAR ROCHA VASCONCELOS	SEMESTRE LETIVO

NOME:

Lista 02 – Alocação Dinâmica e Estruturas

1) Faça um programa simples que peça o nome completo do usuário (máximo 40 caracteres) e possa alocar dinamicamente esta cadeia de caracteres em memória no tamanho exato ao número de caracteres que compõem a cadeia digitada na entrada. Finalmente, o programa deve imprimir esta cadeia utilizando apontadores. Não esqueça de liberar a memória alocada no final!

2) Defina uma estrutura que modele um novo tipo TCliente do mundo real. Use typedef para agilizar este processo. Esta estrutura deve conter: codigo, nome, idade, salario, rua, bairro e cep. Faça com que o programa receba estes dados do usuário e atribua as entradas digitadas por ele em cada um dos campos da estrutura (cuidado com estouro de capacidade dos vetores! O programa deve atribuir apenas a capacidade suportada pelos campos). Ao final, crie uma função imprimeCliente que percorra a estrutura imprimindo cada um dos campos usando apontadores.

3) Vamos realizar modificações no programa anterior: (a) TCliente agora deve ter como um dos campos uma sub-estrutura aninhada TEndereco, a qual deve ser formada pelos campos rua, bairro e cep. Nesta questão, crie a estrutura usando alocação dinâmica. Considerando a definição do tipo acima, novas funções devem ser inseridas no sistema. Veja os novos protótipos de funções:

   a) TCliente* lerCliente(void) (leitura deve ser feita dentro da função)

   b) void imprimeCliente(TCliente *cliente) (o código ficará semelhante ao da função imprimeCliente da questão 2, mas, agora, você deve considerar a estrutura aninhada TEndereço e seus sub-campos na impressão, certo?)

   c) void alterarEndereço(TCliente* cliente, char* rua, char*bairro, char*, unsigned int cep)

   d) void destroiCliente(TCliente *cliente) (desalocar memória para a estrutura criada)