Questão 1.

No contexto da linguagem C, o uso de ponteiros permite maior controle sobre a manipulação de memória. Sobre o uso e as características dos ponteiros, assinale a alternativa correta:

A) Ponteiros armazenam valores binários que são automaticamente convertidos em inteiros.
B) Um ponteiro aponta diretamente para uma posição de memória, permitindo acesso indireto ao conteúdo armazenado.
C) Os ponteiros são exclusivos para manipulação de arquivos e não podem ser usados com variáveis comuns.
D) A principal função dos ponteiros é duplicar variáveis automaticamente em tempo de execução.
E) Ponteiros não podem ser utilizados com arrays, pois são tipos incompatíveis.

Questão 2.

Durante o desenvolvimento de aplicações em C que fazem uso de memória dinâmica, é comum a alocação de estruturas via `malloc`. Qual das opções abaixo representa corretamente a alocação dinâmica de uma estrutura `struct aluno`?

```
struct aluno {
   int matricula;
   float nota;
};
```

A) struct aluno *a = (struct aluno *) malloc(sizeof(struct aluno));
B) struct aluno a = malloc(struct aluno);
C) struct aluno *a = (aluno *) malloc(aluno);
D) struct aluno a = (struct aluno) malloc(sizeof(a));
E) malloc(sizeof(struct aluno));

Questão 3.

Considere o trecho de código abaixo, que utiliza aritmética de ponteiros:

```
#include <stdio.h>

int main() {
    int valores[] = {2, 4, 6, 8, 10};
    int *p = valores + 3;
    printf("%d", *(p - 1));
    return 0;
}
```

Qual será a saída do programa?

A) 2
B) 4
C) 6
D) 8
E) Erro de compilação

Questão 4.

Considere uma estrutura de dados do tipo pilha e as seguintes operações aplicadas a uma pilha “q” inicialmente vazia:

1. PUSH(q, 7)
2. PUSH(q, 14)
3. PUSH(q, 21)
4. POP(q)
5. PUSH(q, 28)
6. PUSH(q, 35)
7. POP(q)
8. PUSH(q, 42)

Após a execução dos comandos acima, qual será o valor no topo da pilha e a soma total dos elementos da pilha?

A) 42 e 91
B) 28 e 49
C) 14 e 63
D) 42 e 77
E) 35 e 84

Questão 5.

A pilha é uma estrutura de dados do tipo LIFO (Last In, First Out), amplamente utilizada em situações como chamadas de funções, algoritmos de backtracking e análise de expressões. Considere que uma pilha “s” está inicialmente vazia e são realizadas as seguintes operações:

1. PUSH(s, 3)
2. PUSH(s, 6)
3. POP(s)
4. PUSH(s, 9)
5. PUSH(s, 12)
6. PUSH(s, 15)
7. POP(s)
8. PUSH(s, 18)
9. PUSH(s, 21)
10. POP(s)

Após todas essas operações, qual será o elemento no topo da pilha “s” e a soma dos elementos da pilha?

A) 21 e 42
B) 18 e 42
C) 15 e 30
D) 12 e 24
E) 9 e 27

Questão 6.

Considere o seguinte código, que utiliza ponteiros e aritmética de ponteiros:

```
#include <stdio.h>

void modificar(int *x, int *y) {
    *x = *x + *y;
    *y = *x - *y;
    *x = *x - *y;
}

int main() {
    int a = 5, b = 7;
    int *p1 = &a, *p2 = &b;
    modificar(p1, p2);
    printf("%d %d", a, b);
    return 0;
}
```

Qual será a saída do programa?

A) 5 7
B) 7 5
C) 12 5
D) 5 12
E) 7 12

Questão 7.

Considere o código abaixo, que aloca dinamicamente memória para um vetor de inteiros:

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

int main() {
    int *arr, n = 5;
    arr = (int *) malloc(n * sizeof(int));
    if (arr == NULL) {
        printf("Erro na alocação de memória");
        return 1;
    }
    for (int i = 0; i < n; i++) {
        arr[i] = i * 2;
    }
    free(arr);
    arr[2] = 10;  // Acesso a memória já liberada
    printf("%d", arr[2]);
    return 0;
}
```

Qual será a saída do programa?

A) 4
B) 10
C) Erro de compilação
D) Valor indefinido
E) 0

Questão 8.

Considere o seguinte código que usa ponteiros para acessar elementos de um array de inteiros:

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int *p = arr;
    p += 3;
    printf("%d", *(p - 2));
    return 0;
}
```

Qual será a saída do programa?

A) 1
B) 3
C) 4
D) 2
E) 5

Questão 9.

Considere as seguintes operações em uma pilha p, inicialmente vazia, e um código que manipula essas operações:

```
PUSH(p, 10)
PUSH(p, 20)
PUSH(p, 30)
POP(p)
PUSH(p, 40)
POP(p)
PUSH(p, 50)
POP(p)
PUSH(p, 60)
```

Após a execução dessas operações, qual será o valor no topo da pilha e a soma dos elementos na pilha?

A) 60 e 40
B) 50 e 40
C) 60 e 130
D) 20 e 130
E) 60 e 90

Questão 10.

Considere o código que utiliza ponteiros e alocação dinâmica de memória para uma estrutura `Aluno`:

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

struct Aluno {
    int matricula;
    float nota;
};

int main() {
    struct Aluno *a = (struct Aluno *) malloc(sizeof(struct Aluno));
    if (a == NULL) {
        printf("Erro na alocação de memória\n");
        return 1;
    }
    a->matricula = 1234;
    a->nota = 8.5;
    printf("Matricula: %d, Nota: %.2f\n", a->matricula, a->nota);
    free(a);
    return 0;
}
```

Qual será a saída do programa?

A) Erro de compilação
B) Matricula: 1234, Nota: 8.5
C) Matricula: 1234, Nota: 0
D) Matricula: 0, Nota: 8.5
E) Nenhuma saída