۱. **الف) (۱۰ نمره)** تابع زیر را با استفاده از زبان C به گونه‌ای کامل کنید که مجموع درایه‌های قطر اصلی و فرعی ماتریس را نمایش دهد. ورودی‌های تابع شامل ابعاد یک ماتریس مربعی (N) و خود ماتریس (matrix[N][N]) می‌باشد. (فقط مجاز به استفاده از نشانگر هستید)

```
int calculateAndPrintDiagonalSums(int N, int matrix[N][N]) {
…
}
```

ب) (۸ نمره)حال کد C تابعی را بنویسید ماتریس مربعی و تعداد سطرهای آن را دریافت می‌کند و اگر ماتریس قطری بود، true و در غیر اینصورت false را برگرداند. (فقط مجاز به استفاده از نشانگر هستید)

```
bool IsDiagonalMatrix(int N, int matrix[N][N]) {
…
}
```

پاسخ:
الف)
راه حل اول:

```
int calculateAndPrintDiagonalSums(int N, int matrix[N][N]) {
    int mainDiagonalSum = 0;
    int secondaryDiagonalSum = 0;

    for (int i = 0; i < N; i++) {                    // (1 point)
        mainDiagonalSum += *(*(matrix + i) + i);     // Sum main diagonal (4 points)
        secondaryDiagonalSum += *(*(matrix + i) + (N - 1 - i));  // Sum secondary diagonal (4
points)
    }

    // Print the diagonal sums  // (1 point)
    printf("Sum of main diagonal elements: %d\n", mainDiagonalSum);
    printf("Sum of secondary diagonal elements: %d\n", secondaryDiagonalSum);
}
```

راه حل دوم:

```
int calculateAndPrintDiagonalSums(int N, int *matrix) {
    int mainDiagonalSum = 0;
    int secondaryDiagonalSum = 0;

    for (int i = 0; i < N; i++) {                    // (1 point)
        mainDiagonalSum += *(matrix + i * N + i);  // Sum main diagonal (4 points)
        secondaryDiagonalSum += *(matrix + i * N + (N - 1 - i));  // Sum secondary diagonal
(4 points)
    }

    // Print the diagonal sums  // (1 point)
    printf("Sum of main diagonal elements: %d\n", mainDiagonalSum);
    printf("Sum of secondary diagonal elements: %d\n", secondaryDiagonalSum);

}
```

راه حل سوم:

```
int calculateAndPrintDiagonalSums(int N, int matrix[N][N]) {
    int mainDiagonalSum = 0;
    int secondaryDiagonalSum = 0;

    // Cast the 2D array to a single pointer for traversal
    int *ptr = (int *)matrix;    //(2 points)


    for (int i = 0; i < N; i++) {                    // (1 point)
        mainDiagonalSum += *(ptr + i * N + i);        // Sum main diagonal (3 points)
        secondaryDiagonalSum += *(ptr + i * N + (N - 1 - i));  // Sum secondary diagonal (3
points)
    }

    // Print the diagonal sums  // (1 point)
    printf("Sum of main diagonal elements: %d\n", mainDiagonalSum);
    printf("Sum of secondary diagonal elements: %d\n", secondaryDiagonalSum);
}
```

ب)
راه حل اول:

```
bool IsDiagonalMatrix(int N, int matrix[N][N]) {
    for (int i = 0; i < N; i++) {                    // (1 point)
        for (int j = 0; j < N; j++) {                // (1 point)
            if (i != j && *(*(matrix + i) + j) != 0) {      // (5 point)
                return false;  // (0.5 point)
            }
        }
    }
    return true;  // (0.5 point)
}
```

راه حل دوم:

```
bool IsDiagonalMatrix(int N, int *matrix) {
    for (int i = 0; i < N; i++) {           // (1 point)
        for (int j = 0; j < N; j++) {       // (1 point)
            if (i != j && *(matrix + i * N + j) != 0) {     // (5 point)
                return false;  // (0.5 point)

            }
        }
    }
    return true;    // (0.5 point)
}
```

راه حل سوم:

```
bool IsDiagonalMatrix(int N, int matrix[N][N]) {
    int *ptr = (int *)matrix;  // Cast the 2D array to a single pointer

    for (int i = 0; i < N; i++) {           // (1 point)
        for (int j = 0; j < N; j++) {       // (1 point)
            if (i != j && *(ptr + i * N + j) != 0) {  // (5 point)
                return false;  // (0.5 point)
            }
        }
    }
    return true;  // (0.5 point)
}
```

۲. (۱۰ نمره) خروجی کد زیر چیست؟ (به صورت کامل توضیح دهید)

```
#include <stdio.h>
void function(char**);

int main() {
    char *arr[] = {"icsp", "fall", "1403", "hw", "6", "pointers"};
    function(arr);
    return 0;
}

void function(char **ptr) {
    char *ptr1;
    ptr1 = (ptr += sizeof(int))[-2];
    printf("%s\n", ptr1);
}
```

پاسخ:

خروجی: 1403

سایز متغیر int برابر ۴ بایت می‌باشد پس عبارت ptr += sizeof(int) به رشته پنجم ارایه یعنی "6"

اشاره دارد زمانیکه عبارت [-2] به آن اضافه می‌شود (ptr += sizeof(int))[-2] یعنی به دوخانه عقب‌تر از خانه پنجم، یعنی خانه سوم اشاره می‌شود و خروجی برابر خانه سوم ارایه (1403) می‌شود.

۳. **(۱۵ نمره)** تابع زیر را به زبان C به صورتی کامل کنید که ارایه مرتب شده‌ی موجود در ادرس addr و با اندازه size* را دریافت کند و عدد new_num را در جای مناسب خود اضافه کند و همچنین سایز ارایه را نیز افزایش دهد.

```
void insert(int *addr, int *new_num, int *size) {
...
}
```

پاسخ:

```
void insert(int *addr, int *new_num, int *size) {
    int pos, i;
    for (i = 0; i < *size; i++)                // 5 points
        if (*new_num < *(addr + i))
            pos = i;

    pos -= 1;
    *size = *size + 1;                         // 2 points

    for (i = *size; i >= pos; i--) {.          // 5 points
        *(addr + i) = *(addr + (i - 1));
    }

    pos -= 1;
    *(addr + pos) = *new_num;                  // 3 points
}
```

۴. **الف) (۶ نمره)** مفهوم pointer to pointer را با ذکر مثال توضیح دهید.
ب) (۴ نمره) دو مورد از مزایا و دو مورد از معایب pass by reference را بیان کنید.

پاسخ:
الف)
این نوع اشاره‌گر آدرس یک اشاره‌گر دیگر را به جای داده‌ها نگه می‌دارد.
این مفهوم به شما این امکان را می‌دهد که به طور غیرمستقیم به داده‌ها از طریق چندین سطح از اشاره‌گرها دسترسی پیدا کنید.

```
int num = 10;          // A regular integer variable
int *ptr = &num;       // Pointer to an integer, stores address of num
int **ptrToPtr = &ptr; // Pointer to a pointer, stores address of ptr
```

ب) مزایا: اشغال کردن کمتر حافظه – تغییر برخی متغیرها از درون توابع
معایب: کم شدن خوانایی کد – اگر در مدیریت reference ها دچار خطا شویم. تغییرات در بعضی مقادیر ممکن است قسمت‌های دیگر برنامه که به متغیرهای تغییر یافته وابسته هستند را دچار مشکل کند.

۵. **امتیازی** (۱۶ نمره) تابع زیر را به گونه‌ای تکمیل کنید که ابتدا اندازه و خود دو آرایه از قبل مرتب شده را دریافت کند و این دو ارایه را با هم ترکیب و مرتب کند و ارایه مرتب و ترکیب شده را بازگرداند. توجه داشته باشید فرایند مرتب کردن ارایه نهایی باید همزمان با ترکیب کردن انجام گیرد (یعنی مرتب کردن ارایه جدید پس از ادغام کامل دو ارایه مجاز نیست). همچنین برای حل این سوال از نشانگر استفاده کنید.

```
Int* mergeAndSortArrays(int *array1, int N1, int *array2, int N2) {
…
}
```

پاسخ:

```
int* mergeAndSortArrays(int *array1, int N1, int *array2, int N2) {

    int *mergedArray = (int *)malloc((N1 + N2) * sizeof(int)); // allocate memory(2 Points)
    if (mergedArray == NULL) {
        printf("Memory allocation failed!\n");
        exit(1); // Exit if memory allocation fails
    }

    int i = 0, j = 0, k = 0;

    // Merge and sort.  (9 Points)
    while (i < N1 && j < N2) {
        if (*(array1 + i) <= *(array2 + j)) {
            *(mergedArray + k) = *(array1 + i);
            i++;
        } else {
            *(mergedArray + k) = *(array2 + j);
            j++;
        }
        k++;
    }

    // Copy remaining elements of array1 (2 Points)

    while (i < N1) {
        *(mergedArray + k) = *(array1 + i);
        i++;
        k++;
    }

    // Copy remaining elements of array2 (2 Points)
    while (j < N2) {
        *(mergedArray + k) = *(array2 + j);
        j++;
        k++;
    }

    return mergedArray; // Return the merged array (1 Points)

}
```