- خروجي قطعه كد زير چيست؟ نحوه توليد خروجي را با ترسيم شكل حافظه شرح دهيد.

```
int j = 0, i = 0, *p1, **p2, **p3;

p2 = p3 =(int **)malloc(3 * sizeof(int *));

do{
   *p2 = (int *)calloc(2 * (i + 1) , sizeof(int));

     for(j = 1, p1 = (*p2) + 1; j < 2 * (i + 1); j++, p1++)
          *p1 += j + *(p1 - 1);

     p2++;
     i++;
}
while(i < 3);
p2 = p3;

for(i = 2; i >= 0; i--)
     for(j = 0; j < 2 * (i + 1); j++)
          printf("[%d][%d] = %d\n", i, j, *((*(p2 + i))+j));
```

- فرض كنيد اطلاعات زمان در struct time به صورت زير ذخيره مي‌شود. تابع add_time را كامل كنيد. اين تابع دو زمان را مي‌گيرد و جمع آنها را برمي‌گرداند (ثانيه و دقيقه نمي‌توانند بيشتر از ۶۰ باشند)

```
struct time{
   int h, m, s;
};

struct time add_time(struct time t1, struct time t2)
```

- تابعي بنويسيد كه عدد n و اعداد m_1، m_2 ... m_n را از ورودي بخواند و يك ماتريس دو بعدي درست كند كه بعد اول آن n است ولي اندازه بعد دوم آن براي هر سطر متغير و برابر m_i است.

براي مثال اگر $n = 3$, $m_1 = 4$, $m_2 = 2$, $m_3 = 5$ باشد، آرايه زير توليد مي‌شود.

4	4	4	4	
2	2			
5	5	5	5	5

```
int **matrix;
printf("Enter n: ");
scanf("%d", &n);

matrix = (int **)calloc(n, sizeof(int *));

int i, j;
for(i = 0; i < n; i++){
      printf("Enter m: ");
      scanf("%d", &m);

      matrix[i] = (int *)calloc(m, sizeof(int));

      for(j = 0; j < m; j++)
            matrix[i][j] = m;

}
```

- تابع tafazol را بنويسيد، اين تابع دو مجموعه را كه با linked-list پياده‌سازي شده است را مي‌گيرد و A – B را به صورت يك linked-list برمي‌گرداند (در اين سوال هم از struct time استفاده شده است)

```
struct node{
   struct time t;
   struct node * next;
}

struct node * tafazol(struct node * A, struct node * B)
```

- برنامه‌اي بنويسيد كه اسم يك فايل و دو عدد n و m را بگيرد و خطهاي n، n+1، n+m ... n+2 را چاپ كند. در صورتي كه شماره خطهاي مورد نظر در فايل وجود نداشته باشد، پيغام مناسب دهد.

- تابعي بنويسيد كه يك رشته را بگيرد و حروف بزرگ و كوچك آنرا تصحيح كند. در رشته تصحيح شده، هر كلمه با حرف بزرگ شروع مي‌شود و ساير حروف كوچك است. براي تبديل به حرف كوچك و بزرگ مي‌توانيد از تابع‌هاي زير استفاده كنيد

```
#include <ctype.h>
char tolower(char c)
char toupper(char c)
```

براي مثال

```
THIS is Test      → This Is Test
thiS iS TesT      → This Is Test
```

الگوريتم كار به اين صورت است كه دو حرف رشته را بررسي مي‌كنيم، يكي newc است و ديگري oldc. newc حرف فعلي است و oldc حرف قبلي. اگر حرف فعلي جاي خالي نباشد و حرف قبلي جاي خالي باشد بنابراين اين حرف اول كلمه است و toupper صدا زده مي‌شود. در صورتي كه هردو حرف جاي خالي نباشد بنابراين حرف وسط كلمه است و tolower صدا زده مي‌شود. اولا دقت كنيد كه toupper اگر حرف بزرگ باشد كاري نمي‌كند و به صورت مشابه اگر tolower براي حروف كوچك صدا زده شود تاثيري ندارد. دوما براي اينكه حرف اول اولين كلمه در رشته به درستي تبديل به حرف بزرگ بشود ما مقدار اوليه oldc را برابر جاي خالي قرار داده‌ايم.

```
char *correct(char *s){
    int i = 0;
    char oldc, newc;
    oldc = ' ';
    do{
        newc = s[i];

        if((newc != ' ') && (oldc == ' '))
            s[i] = toupper(s[i]);
        if((newc != ' ') && (oldc != ' '))
            s[i] = tolower(s[i]);

        oldc = newc;
        i++;
```

```
    }while(s[i] != '\0');

    return s;
}
```

- برنامه‌اي بنويسيد كه اسم يك فايل را بگيرد و تعداد تكرار ارقام 0 ... 9 در آن فايل را چاپ كند.

- فرض كنيد اطلاعات چندين نقطه با استفاده از struct زير در يك فايل باينري ذخيره شده است.

```
struct point{
    int x, y;
};
```

الف) تابعي بنويسيد كه دو عدد i و j و يك File Handler را بگيرد و نقطه i و j را باهم جابجا كند.

```
void swap(FILE *fp, int i, int j){
    struct point p1, p2, tmp;

    fseek(fp, i * sizeof(p1), SEEK_SET);
    fread(&p1, sizeof(p1), 1, fp);

    fseek(fp, j * sizeof(p2), SEEK_SET);
    fread(&p2, sizeof(p2), 1, fp);

    tmp = p1;
    p1 = p2;
    p2 = tmp;

    fseek(fp, i * sizeof(p1), SEEK_SET);
    fwrite(&p1, sizeof(p1), 1, fp);

    fseek(fp, j * sizeof(p2), SEEK_SET);
```

```
        fwrite(&p2, sizeof(p2), 1, fp);
}
```

ب) تابعي بنويسيد كه دو نقطه را بگيرد. اگر نقطه اول بزرگتر بود ١، اگر دومي بزرگتر بود ١- و اگر برابر بودند ٠ برگرداند. ترتيب بين نقاط به اين صورت تعريف مي‌شود:

$$if(\sqrt{x_1^2 + y_1^2} > \sqrt{x_2^2 + y_2^2}) \text{ ® } p_1 > p_2.$$

```
int point_cmp(struct point p1, struct point p2){
    double pd1 = sqrt(pow(p1.x, 2) + pow(p1.y, 2));
    double pd2 = sqrt(pow(p2.x, 2) + pow(p2.y, 2));

    if(pd1 < pd2)
          return -1;
    else if(pd1 > pd2)
         return 1;
    else
        return 0;
}
```

ج) تابعي بنويسيد كه يك File Handler و عدد i را بگيرد و محل بزرگترين نقطه از نقطه i تا انتهاي فايل را برگرداند

```
int get_max_index(FILE *fp, int i){
    int max_index, index = i;
    struct point max_val, tmp;

    max_index = i;
    fseek(fp, i * sizeof(struct point), SEEK_SET);
    fread(&max_val, sizeof(max_val), 1, fp);
    index++;

    while(fread(&tmp, sizeof(max_val), 1, fp) == 1){
        if(point_cmp(tmp, max_val) > 0){
```

```
            max_index = index;
            max_val = tmp;
        }
        index++;
    }

    return max_index;
}
```

د) با استفاده از تابع‌هايي كه در مراحل الف و ب و ج تعريف شده است. يك برنامه كامل C بنويسيد كه اسم يك فايل باينري را از كاربر بگيرد و آنرا به صورت نزولي مرتب كند. براي مرتب كردن از آرايه استفاده **نكنيد**.