1. داخل آرایه ای از کاراکتر به نام ArrayA به طول n الگویی نوشته شده است. همچنین آرایه دیگری به نام ArrayB به طول m داریم که حروفی در آن قرار دارد. که m از n بزرگ تر است.

تابع زیر را طوری کامل کنید که در صورت وجود الگوی ArrayA در آرایه ArrayB مقدار 1 و در غیر این صورت مقدار 0 را باز گرداند.
مثال:

ArrayA = {A, B, A, A}
ArrayB = {C,D,A,F,A,B,A,A,P,R}
return value: 1

چون داخل ArrayB دقیقا کاراکترهای داخل ArrayA با همان ترتیب تکرار شده است.

دقت کنید که محتوای هیچ یک از این دو آرایه string نیست.

```
int containsPattern(char ArrayB[], int m, char ArrayA[], int n){

}
```

2. میخواهیم خانه های یک شهر را پلاک گذاری کنیم به صورتی که هیچ دو خانه ای نباشد که پلاک یکسانی داشته باشند. برای این کار از یک لیست پیوندی (linked list) کمک می‌گیرم که در هر node آن، یک اشاره گر به خانه بعدی و یک عدد int که بیانگر پلاک است وجود دارد.

الف- struct هر عضو از این لیست پیوندی را بنویسید. (5 نمره)

```
typedef struct Node{

} Node;
```

ب- تابع زیر را طوری بنویسید که یک اشاره گر به اول لیست پیوندی و یک عدد که پلاک جدید است را دریافت میکند و در صورتی که آن پلاک تا به حال ثبت نشده بود، آن را به آخر لیست اضافه کند و در خروجی اشاره گر به سر لیست پیوندی را بازگرداند. (5 نمره)

```
Node* appendNode(Node* headRef, int plate){
```

}

ج- حال میخواهیم پلاک خانه هایی که خراب شده اند را از لیست حذف کنیم. تابع زیر را طوری کامل کنید که یک اشاره گر به اول لیست پیوندی و یک عدد که پلاکی است که باید حذف شود را در ورودی دریافت کرده و در صورت وجود آن عدد در لیست، آن node را حذف کند. (10 نمره)

```
void deleteNode(Node* headRef, int plate){

}
```

د- حال تابع زیر را طوری بنویسید که در ورودی اشاره گر به اول لیست پیوندی را گرفته و لیست را به شکلی مرتب سازی کند که پلاک ها به صورت صعودی قرار بگیرند. (10 نمره)

```
void sortList(Node* headRef){

}
```

ه- حال میخواهیم لیست را به شکلی پیاده سازی کنیم که همواره پلاک ها به صورت صعودی داخل لیست مرتب باشند و نیازی نباشد که تابعی برای مرتب سازی آن بنویسیم. برای این کار کافی است که تابع اضافه کردن پلاک که در بند ب نوشتید را تغییر دهیم. تابع appendNode را به شکلی بنویسید که خواسته ما برآورده شود و هر نود ورودی به جای اضافه شدن به انتهای لیست پیوندی در جایی که اضافه شود که لیست همواره به ترتیب صعودی مرتب باشد. (10 نمره)

3. در یک خیابان تعدادی تاکسی و مسافر وجود دارد. هر تاکسی باید مسافری را سوار کند که فاصله آن با او کمتر از d باشد. به دنبال آن هستیم حداکثر مسافر را بتوانیم سوار تاکسی کنیم. آرایه ای از int به طول n داریم که نشان دهنده خیابان است. تاکسی را با 2 و مسافر را با 1 در آرایه مشخص می کنیم و همچنین اگر مقدار خانه ای 0 باشد یعنی هیچ چیز در آن وجود ندارد. همچنین فاصله یک تاکسی با یک مسافر برابر است با تفاضل index آن ها در آرایه است. تابع زیر را به شکلی کامل کنید که در آن بیشترین تعداد مسافری که میتوانند سوار تاکسی شوند را به دست آورد و بازگرداند.

```
int maxPassenger(int avenue [], int n, int d){

}
```

مثال:

D = 2

2	1	1	0	0	2	0	1	1	1	1	2	0	0	2	0	1	2

در این حالت بیشترین تعداد مسافری که میتوان سوار بر تاکسی شوند برابر با 4 است.

⚠️ در تحویل تمرین، حتما به نکات زیر توجه نمایید:

- ❖ انجام این تمرین اختیاری است، ولی با وجود آن اکیداً توصیه می کنیم که برای آمادگی بیشتر برای امتحان و همچنین کمکی که می تواند برای پروژه سوم شما داشته باشد، این تمرین اختیاری را هم مشابه تمرین های عادی تا 1403/10/18 انجام دهید.
- ❖ نیازی به بارگذاری این تمرین در سامانه Elearn نیست.
- ❖ پس از مطالعه کامل تمرین، در صورت هرگونه ابهام با پارسا ناصری در ارتباط باشید.
  - موضوع ایمیل را *HW8 ICSP* وارد نمایید.