۱. **(۷ نمره)**

الف-(۴ نمره) الگوریتمی را با استفاده از فلوچارت نمایش دهید که تعداد ارقام یک عدد صحیح غیر منفی را شمرده و روی صفحه چاپ نماید.

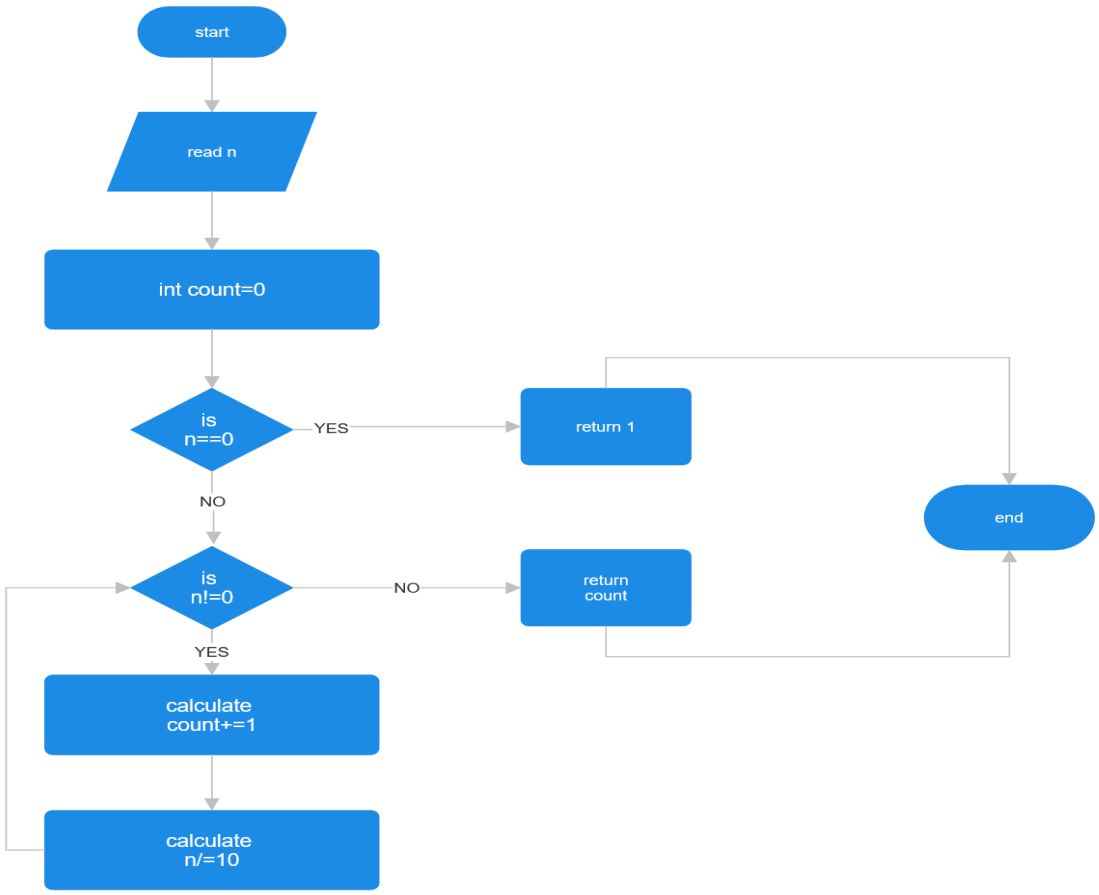
ب-(۳ نمره) با استفاده از الگوریتم بند الف، تابع زیر را به زبان C طوری بنویسید که تعداد ارقام عدد صحیح غیر منفی n را، که به عنوان آرگومان ورودی دریافت می‌کند، شمارش نموده و نتیجه را در قالب مقدار برگشتی بازگرداند.

```
int NumofDigits(int n){

}
```

- پاسخ

فلوچارت:4 نمره



کد زبان c: 3 نمره

```c
int NumofDigits(int n) {
    int count = 0;
    while (n > 0) {
        count++;
        n /= 10;
    }
    return count;
}
```

۲. **(۶ نمره)**

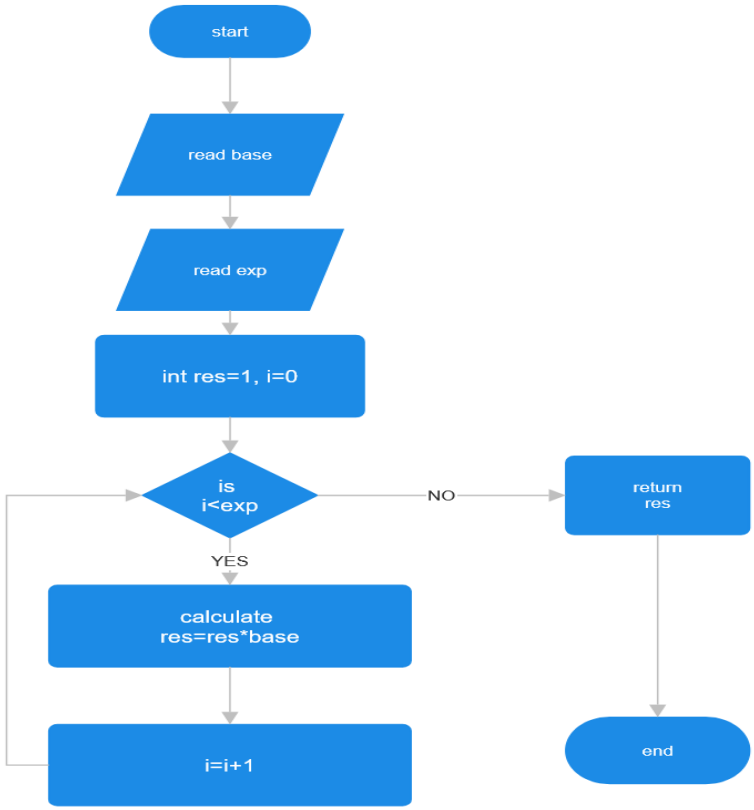
الف–(۴ نمره) الگوریتمی را با استفاده از فلوچارت نمایش دهید که دو عدد را از کاربر دریافت نموده و عدد اول را به توان عدد دوم برساند. در این الگوریتم باید توان را پیاده‌سازی کنیم.

ب–(۳ نمره) بر مبنای الگوریتم بند الف، تابع زیر را به زبان C طوری بنویسید که دو آرگومان ورودی base و exp را دریافت کرده و آرگومان اول را به توان آرگومان دوم برساند و حاصل را در قابل مقدار بازگشتی برگرداند. در نوشتن این تابع نمی‌توانیم از تابع کتابخانه‌ای pow() برای توان استفاده کنیم و باید مشابه بند الف توان را با یک حقه پیاده‌سازی کنیم.

```
int PowerCalculator(int base,int exp){
}
```

- پاسخ:

فلوچارت:4 نمره



کد زبان c: 3نمره

```c
int PowerCalculator(int base, int exp) {
    int res = 1;//any number to the power of zero is one
    for (int i = 0; i < exp; i++) {
        res =res * base;
    }
    return res;
}
```

۳. (۱۳ نمره)

الف- (۳ نمره) در سوال ۶ تمرین دوم با الگوریتم اعداد Armstrong آشنا شدید. این الگوریتم را با استفاده از دو تابع سوال های ۱ و ۲ این تمرین در قالب یک فلوچارت بازنویسی کنید.

ب-(۱۰ نمره) با استفاده از توابع سوال ۱ و ۲ تابع زیر را در زبان C طوری کامل کنید که مشخص کند عدد n که به عنوان آرگومان دریافت می‌کند Armstrong است یا خیر. در صورتی که عدد Armstrong است مقدار یک و در غیر این صورت مقدار صفر را برگرداند.

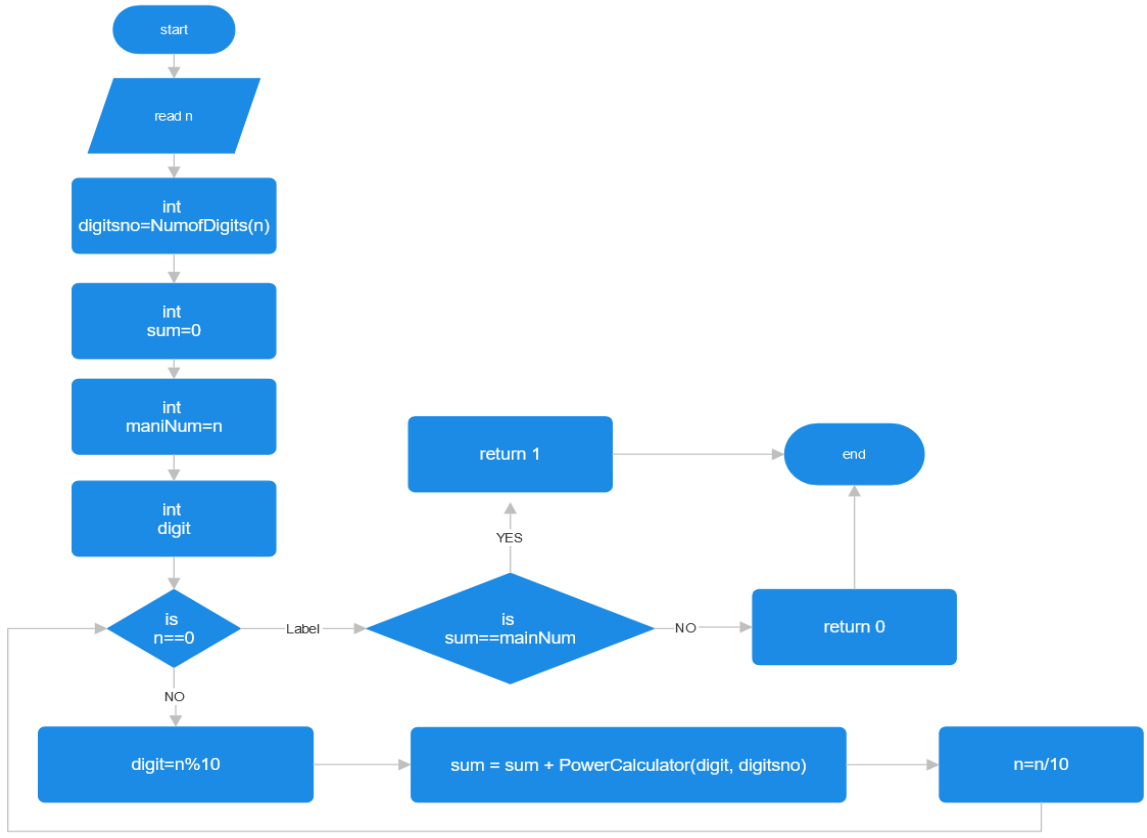
```
int isArmstrong(int n){
}
```

- پاسخ: فلوچارت 3 نمره

نمره دهی کد C: استفاده ی صحیح از توابع سوال 1: 5 نمره

حلقه ی while : 3 نمره

باقی موارد: 2نمره



```c
int isArmstrong(int n) {
    int digitsno = NumofDigits(n);
    int sum = 0;
    int mainNum = n;//for further comparison
    while (n != 0) //for calculating the sum of each digit raised to the power of number of digits
    {
        int digit = n % 10;
        sum += PowerCalculator(digit, digitsno);
        n /= 10;
    }
    return (sum == mainNum);
}
```

4. (**2 نمره**) دو مورد از اشکالات استفاده از void function را توضیح دهید.

- پاسخ: 1- به علت اینکه توابع void مقداری را بازنمیگردانند در نتیجه در زمان debug کردن کدها به مشکل برمیخوریم.

2-استفاده از توابع void باعث میشود مقدار خروجی بازگردانده نشود و در نتیجه ما از یکی از اهداف اصلی توابع که عدم تکرار کد در برنامه بود دور میشویم. مثلا اگر تابعی void داشته باشیم که ضرب دو آرگومان ورودی را محاسبه کرده سپس مقدار ضرب را print کند و ما در جای دیگری از برنامه نیاز به ضرب دو عدد داشته باشیم مجد باید کدی جداگانه بنویسیم و نمیتوانیم از تابع قبل استفاده کنیم.

تابع مثال زده شده:

```c
void mult(int a, int b) {
    int mult_res = a * b;
    printf("%d\n", mult_res);
}
```

5. **سوال امتیازی ** (15 نمره)**

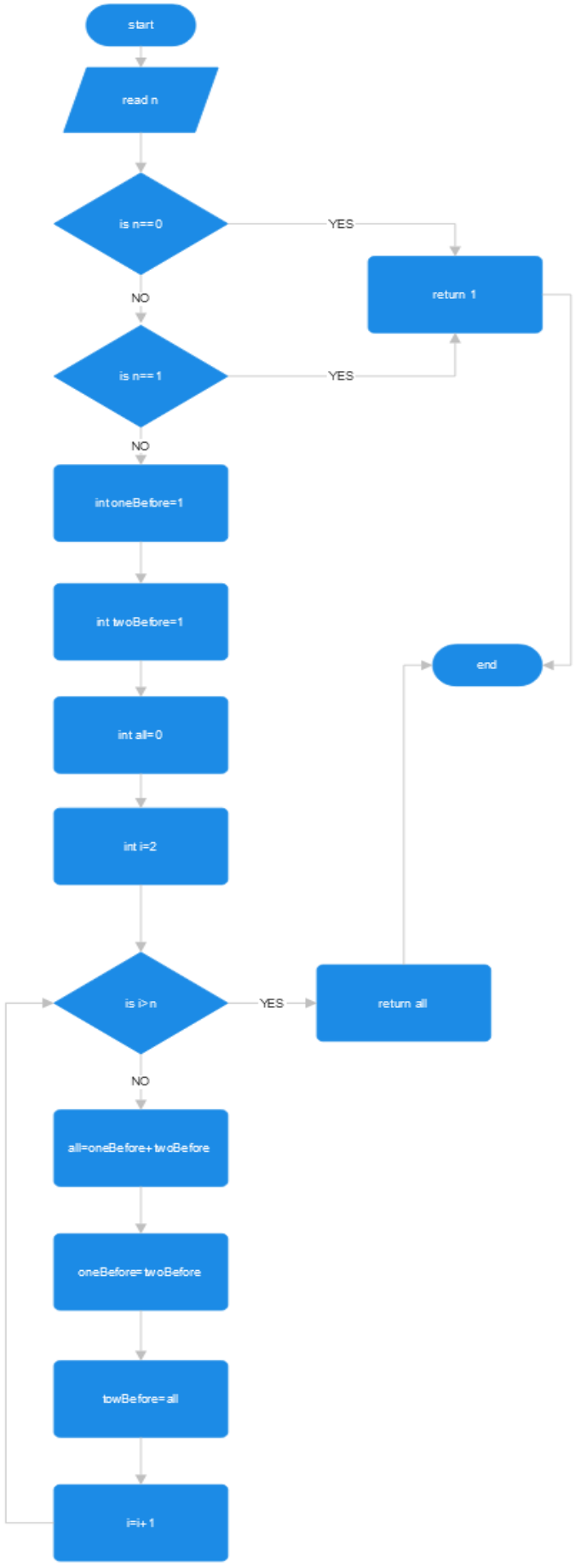
قصد داریم با رد شدن از چند مانع از نقطه a به b برویم. اینکه با چند مانع روبرو هستیم را کاربر مشخص می‌کند. اگر فرض کنیم تنها می‌توانیم با هر پرش 1 یا حداکثر دو مانع را پشت سر بگذاریم، می‌خواهیم ببینیم به چند طریق میتوانیم از a به b برسیم.
اگر کاربر عدد صفر را انتخاب نمود یک راه برای رسیدن به مقصد داریم.

مثال: تعداد موانع=3
حالت اول: با سه پرش که در هر پرش از یک مانع عبور می‌کنیم به مقصد برسیم.
حالت دوم: پرش اول از یک مانع و پرش دوم از دو مانع عبور کنیم.
حالت سوم: پرش اول از دو مانع و پرش دوم از دو مانع عبور کنیم.
پس به ازای n=3 به سه طریق میتوان به مقصد رسید.

الف)(9 نمره) فلوچارت این مساله را پیاده کنید.

پاسخ: الگوریتم با استفاده از روش Iterative پیاده سازی شده است.

توضیح الگوریتم: در صورتی که یک یا صفر مانع داشته باشیم یک راه بیشتر نداریم اما برای تعداد موانع بیشتر الگوریتم به این صورت کار میکند که اگر فرض کنیم 5 مانع بر سر راه داریم در این صورت برای اینکه در نقطه ی بعد از مانع پنجم باشیم دو راه داریم: یا با یک پرش از مانع پنجم رد شده باشیم یا در نقطه ی بعد از مانع سوم بوده ایم و با یک پرش مانع چهارم و پنجم را پشت سر گذاشته باشیم. به همین جهت میبینم چند حالت وجود دارد که ما در نقطه ای بعد از مانع چهارم و در نقطه ی پس از مانع سوم باشیم. این چرخه ادامه پیدا میکند تا به صفر یا یک برسیم که با توجه به توضیحات میدانیم در این حالت ها یک راه بیشتر نداریم.


start
read n
is n==0
YES
return 1
NO
is n==1
YES
NO
int oneBefore=1
int twoBefore=1
int all=0
int i=2
end
is i>n
YES
return all
NO
all=oneBefore+twoBefore
oneBefore=twoBefore
towBefore=all
i=i+1

ب) (6 نمره) الگوریتم قسمت الف را در زبان c پیاده سازی کنید.

```
int NumberofWays(int) {

}
```

```c
int NumberofWays(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    int oneBefore = 1;  // Number of ways to pass the n-1 barriers
    int twoBefore = 1;  // Number of ways to pass the n-2 barriers
    int all = 0;//all the ways posible to pass the n barriers and reach the
end
    for (int i = 2; i <= n; i++) {//starting from i=2 because of the given
facts of the question
        all = oneBefore + twoBefore;
        twoBefore = oneBefore;
        oneBefore = all;
    }
    return all;
}
```