# Exercise 6: Arrays and 2D Arrays

R Ram Kaushik

April 3, 2018

## 1 Define Boolean functions

- `is_prime(n)` that tests whether a non-negative integer `n` is prime or not.
- `is_cube(n)` that tests whether number `n` is a perfect cube.
- `is_divisible_by (n, d)` that tests whether an integer `n` is divisible by integer `d`.

### 1.1 Specification

2 functions `is_prime(),is_cube()` which takes the number `n` as the input, a function `is_divisible()`, which takes 2 numbers `n, d` as the inputs and returns a boolean value to the calling function.

### 1.2 Prototype

```
bool is_prime(int n)
bool is_cube(int n)
bool is_divisible(int n,int d)
```

### 1.3 Program Design

The program consists of 3 functions `is_prime(int n),is_cube(int n),is_divisible(int n,int d)` which checks the condition and returns a value and `main()` which reads the numbers from `stdin` and calls the functions to test it.

### 1.4 Algorithm

```
def is_prime(n):
   flag = true
   for i in range(2,n):
      if n%i==0:
         flag=false
         break
   return flag
def is_cube(n):
```

```
   flag=false
   i=1
   while i*i*i<=n:
      if i*i*i==n:
         flag=true
         break
   return flag
def is_divisible(n,d):
   flag=false
   if n%d==0:
      flag=true
   return flag
```

### 1.5 Source Code

```
#include<stdio.h>
#include<stdbool.h>
bool is_prime(int n){
  int i;
  bool flag=true;
  for(i=2;i<n;i++){
    if (n%i==0){
      flag=false;
      break;
    }
  }
  return flag;
}
bool is_cube(int n){
  int i=1;
  bool flag=false;
  while((i*i*i)<=n){
    if ((i*i*i)==n){
      flag=true;
      break;
    }
    i++;
  }
  return flag;
}
bool is_divisible(int n,int d){
  bool flag=false;
  if (n%d==0){
    flag=true;
  }
  return flag;
```

```
}
int main(){
  int a,b,c,d;
  bool e,f,g;
  scanf("%d%d%d%d",&a,&b,&c,&d);
  g=is_prime(a);
  f=is_cube(b);
  e=is_divisible(c,d);
  printf("%d\n%d\n%d",g,f,e);
}
```

### 1.6 Test Input

```
11
216
56 7
```

### 1.7 Output

1
1
1

## 2 Sorting

Sort the list of numbers based on their weights, where the weight of a number is defined as

$$\text{weight}(n) = \begin{cases} 3 & n \text{ is prime.} \\ 4 & n \text{ is a multiple of 4 and divisible by 6.} \\ 5 & n \text{ is a pefect cube.} \end{cases}$$

### 2.1 Specification

2 functions `is_prime()`, `is_cube()` which takes the number `n` as the input, a function `is_divisible()`, which takes 2 numbers `n`, `d` as the inputs and returns a boolean value to the calling function, `weight_calc()`, which takes arrays `a[]`, `weight[]` and length of array `n` as inputs and assigns values to `weight[]` as per the conditions, `swap()` which takes array `a[]`, and 2 indices `m`, `n` as inputs and swaps the 2 numbers, and `selectionSort()` which takes 2 arrays `arr[]`, `b[]`, length of array `n` as inputs and sorts the array in ascending order.

### 2.2 Prototype

```
bool is_prime(int n)
bool is_cube(int n)
bool is_divisible(int n,int d)
```

```
void weight_calc(int a[],int weight[],int n)
void swap(int a[], int m, int n)
void selectionSort(int arr[],int b[], int n)
```

### 2.3 Program Design

The program consists of 3 functions `is_prime(int n)`,`is_cube(int n)`,`is_divisible(int n,int d)` which checks the condition and returns a value, `weight_calc(int a[],int weight[],int n)` which assigns the values to `weight[]` array based on the condition, `swap(int a[], int m, int n)` which swaps 2 numbers, `selectionSort(int arr[],int b[], int n)` which sorts the array in ascending order and `main()` which reads the numbers from `stdin` and calls the functions to test it and print the result on `stdout`.

### 2.4 Algorithm

```
def is_prime(n):
   flag = true
   for i in range(2,n):
      if n%i==0:
         flag=false
         break
   return flag
def is_cube(n):
   flag=false
   i=1
   while i*i*i<=n:
      if i*i*i==n:
         flag=true
         break
   return flag
def is_divisible(n,d):
   flag=false
   if n%d==0:
      flag=true
   return flag
def weight_calc(a[],weight[],n)
   for i in range(n):
      t=is_prime(a[i])
      u=is_cube(a[i])
      v=is_divisible(a[i],12)
      if t==true:
         weight[i]=3
      elif v==true:
         weight[i]=4
      elif u==true:
         weight[i]=5
      else
```

```
        weight[i]=0
def swap(a[],m,n):
   t=a[m]
   a[m]=a[n]
   a[n]=t
def selectionSort(arr[],b[],n):
   for i in range(n-1):
      m=i
      for j in range(i+1,n):
         if a[j]<a[m]:
            m=j
      swap(arr,m,i)
      swap(b,m,i)
```

### 2.5 Source Code

```
#include<stdio.h>
#include<stdbool.h>
bool is_prime(int n){
  int i;
  bool flag=true;
  for(i=2;i<n;i++){
    if (n%i==0){
      flag=false;
    }
  }
  return flag;
}
bool is_cube(int n){
  int i=1;
  bool flag=false;
  while((i*i*i)<=n){
    if ((i*i*i)==n){
      flag=true;
    }
    i++;
  }
  return flag;
}
bool is_divisible(int n,int d){

  bool flag=false;
  if (n%d==0){
    flag=true;
  }
  return flag;
```

```
}
void weight_calc(int a[],int weight[],int n){
  bool t,u,v,w;
  int i;
  for(i=0;i<n;i++){
    t=is_prime(a[i]);
    u=is_cube(a[i]);
    v=is_divisible(a[i],12);
    if (t==true){
      weight[i]=3;
    }
    else if (v==true){
      weight[i]=4;
    }
    else if (u==true){
      weight[i]=5;
    }
    else{
      weight[i]=0;
    }
  }
}
void swap(int a[], int m, int n){
  int t = a[m];
  a[m]=a[n];
  a[n]=t;
}

void selectionSort(int arr[],int b[], int n){
  int i, j, min_idx;
  for (i = 0; i < n-1; i++){
    min_idx = i;
    for (j = i+1; j < n; j++){
      if (arr[j] < arr[min_idx]){
min_idx = j;
      }
    }
    swap(arr,min_idx,i);
    swap(b,min_idx,i);
  }
}
int main(){
  int a[10]={23,46,42,287,288,164,973,713,94,56};
  int weight[10],b[10],i;
  weight_calc(a,weight,10);
  for(i=0;i<10;i++){
```

```
    printf("%d ",weight[i]);
  }
  selectionSort(weight,a,10);
  printf("\n");
  for(i=0;i<10;i++){
    printf("%d ",weight[i]);
  }
  printf("\n");
  for(i=0;i<10;i++){
    printf("%d ",a[i]);
  }
  return 0;
}
```

### 2.6 Output

```
 3  0   0   0   4   0  0  0  0   0
 0  0   0   0   0   0  0  0  3   4
46 42 287 164 973 713 94 56 23 288
```

## 3 Mean Height

Populate an array `heights[N]` with heights of persons and find how many persons are above the average height.

### 3.1 Specification

A function to find average height and number of people above average.

### 3.2 Program Design

The program consists of `main()`, which reads the input from stdin, finds the average, finds number of people above average height, and prints it on `stdout`.

### 3.3 Algorithm

```
s=0
c=0
for i in range(n):
   s+=a[i]
avg=sum/n
for i in range(n):
   if(a[i]>avg:
      c++
```

### 3.4 Source Code

```
#include<stdio.h>
int main(){
  int i,n,count=0;
  float sum=0,avg,height[100];
  scanf("%d",&n);
  for(i=0;i<n;i++){
    scanf("%f",&height[i]);
    sum=sum+height[i];
  }
  avg=sum/n;
  for(i=0;i<n;i++){
    if(height[i]>avg){
      count++;
    }
  }
  printf("%f\n%d",avg,count);
  return 0;
}
```

### 3.5 Test Input

```
10
172 186 154 123 145 166 169 150 140 177
```

### 3.6 Output

```
158.200005
```

## 4 BMI

Populate a two dimensional array `a[N][N]` with heights and weights of persons and compute the Body Mass Index (BMI) of the individuals. `a[i][0]` and `a[i][1]` are the height and weight of `i` th person. BMI is defined as

$$\text{BMI} = \frac{\text{weight}}{\text{height}^2}$$

where weight is in kg and height is in m.

### 4.1 Specification

A function which calculates the bmi of a person.

### 4.2 Program Design

The program consists of `main()`, which gets the input from `stdin`, finds the bmi and prints the output on `stdout`.

### 4.3 Algorithm

```
for i in range(n):
   bmi[i]=a[i][1]/(a[i][0]*a[i][0])
```

### 4.4 Source Code

```
#include<stdio.h>
int main(){
  int i,j,n;
  float bmi[10],a[10][2];
  scanf("%d",&n);
  for(i=0;i<n;i++){
    for(j=0;j<2;j++){
      scanf("%f",&a[i][j]);
    }
  }
  for(i=0;i<n;i++){
    bmi[i]=(a[i][1]/(a[i][0]*a[i][0]));
    printf("%f\n",bmi[i]);
  }
  return 0;
}
```

### 4.5 Test Input

```
5
1.72 65
1.77 70
1.54 60
1.86 86
1.70 75
```

### 4.6 Output

```
21.971336
22.343515
25.299377
24.858366
25.951555
```