Exercise 6: Arrays and 2D Arrays

PRAVEEN KUMAR R

April 14, 2018

Assignment	6
Reg No	312217104114
Name	PRAVEEN KUMAR R
Grade	
Date	17-03-2018

1 Boolean functions

Define the following 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`.

Test these functions from main and print the return values.

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

1.6.1 Input

31 9790 91 13

1.6.2 Output

1
0
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 algorithm development

- define 3 different function to check whether
  1. the given number is prime or not (`is_prime(n)`)
  2. the given number is divisible by 4 and 6 or not(`is_div(n)`)
  3. the given number is perfect cube or not(`is_cube(n)`)
- define another function to return the sum of all values of the above function when they all are called with a number n.(`result(n)`) -define a funtion sort that calls reuslt fuction for every element in the array and sort the elemets based on this return value.(`sort(a,n)`) -By calling the sort function from main() by passing the array and the elements of the array as the parameters we complete the task of sorting the number.

2.2 functions used

- `is_prime(n)`

– input:an integer n – output:3, if n is prime, else 0

- `is_div(n)`

– input:an integer n – output:4, if n is divisible by 4 and 6, else 0

- `is_cube(n)`

– input:an integer n –output:5, if n is perfect cube, else 0

- `result(n)`

– input:an integer n – output: `is_prime(n)` + `is_div(n)` + `is_cube(n)`

- `sort(a,n)`

– input: a,an integer array n,an integer (number of elements in the array a) – output: sorted array such that `result(a[0])` <= `result(a[1])` <= ..... `result(a[n-1])`

2.3 program

```
#include<stdio.h>
int is_prime(int n)
{
  int f=0;
  int i=1;
  while(i<=n)
    {
      if(n%i==0)
f++;
      i++;

    }
  if(f==2)
    return 3;
  else
    return 0;
}
int is_div(int n)
{
  if(n%4==0&&n%6==0)
    return 4;
  else
    return 0;
}
int is_cube(int n)
{
  int flag=0;
  for(int j=1;j*j*j<=n;j++)
    if(j*j*j==n)
      {
flag=1;
break;
      }
  if(flag)
    return 5;
  else
    return 0;
}
int result(int n)
{
  return is_prime(n)+is_div(n)+is_cube(n);
}
void sort(int a[],int n)
{
  int i,j;
```

```
  int temp;
  for(int i=0;i<n-1;i++)
    {
      int min=i;
      for(int j=i+1;j<n;j++)
if(result(a[j])<result(a[min]))
  min=j;
      temp=a[i];
      a[i]=a[min];
      a[min]=temp;
    }
}
int main()
{
  int a[100];
  int n;
  scanf("%d",&n);
  for(int i=0;i<n;i++)
    scanf("%d",&a[i]);
  for(int i=0;i<n;i++)
    printf("%d %c",a[i],i<n-1?',':'\n');
  sort(a,n);
  for(int i=0;i<n;i++)
    printf("%d : %d\n",a[i],result(a[i]));
}
```

2.4 Test

2.4.1 Input

24 1245 34 55 34 125 86 443

2.4.2 Output

```
24 ,1245 ,34 ,55 ,34 ,125 ,86 ,443
1245 : 0
34 : 0
55 : 0
34 : 0
86 : 0
443 : 3
24 : 4
125 : 5
```

3 BMI calculation

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

3.1 algorithm development

- first define a funtion to find the average of n numbers present in the array.
- once the average is computed each element in the array is compared with the average and if the element is greater than the average then the count of number of such element is increamented

3.2 functions used

`cal_avg`

- input: a, an interger array. n, an integer denoting the number of elements in the array.
- output: average of n elements in the array

3.3 program

```
#include<stdio.h>
float cal_avg(int a[],int n)
{
 float sum=0;
 for(int i=0;i<n;i++)
    sum+=a[i];
 return sum/n;
}
int main()
{
 int a[100],n;
 scanf("%d",&n);
 for(int i=0;i<n;i++)
  scanf("%d",&a[i]);
for(int i=0;i<n;i++)
  printf("%d%c",a[i],i<n-1?',':'\n');
 float avg= cal_avg(a,n);
 int count=0;
 for(int i=0;i<n;i++)
   if(avg<a[i])
      count++;
 printf("\n%d",count);
return 0;
}
```

23,43,56,43,31,24,33,22

3

1. 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.

```
#include<stdio.h>
void calc_bmi(float a[100][2],int n,float bmi[100])
{
  for(int i=0;i<n;i++)
    {
      bmi[i]=a[i][1]/(a[i][0]*a[i][0]);
    }
}
int main()
{
  float a[100][2],bmi[100];
  int n;
  scanf("%d",&n);
  for(int i=0;i<n;i++)
    scanf("%f%f",&a[i][1],&a[i][0]);
  calc_bmi(a,n,bmi);
  printf("weight\theight\tBMI\n");
  for(int i=0;i<n;i++)
    {
      printf("%f \t %f \t %f \n",a[i][1],a[i][0],bmi[i]);
    }
}
```

3.4 Test

3.4.1 Input

```
5
45 1.50
60.1 2.01
89.50 2.2
38.40 1.4
77.20 1.84
```

3.4.2 Output

weight	height	BMI
45.0	1.5	20.0
60.099998	2.01	14.875869
89.5	2.2	18.491735
38.400002	1.4	19.591839
77.199997	1.84	22.802456