

Name: Santhosh M
Reg No: 240701475

Week 3: Decision Making and Branching – if, if...else, nested if...else, if...else if, Switch-Case

1. Same Digit

Problem statement:

Write a program to read two integer values and print true if both the numbers end with the same digit, otherwise print false. Example: If 698 and 768 are given, program should print true as they both end with 8. Sample Input 1 25 53 Sample Output 1 false Sample Input 2 27 77 Sample Output 2 true

Program:

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,c,d;
5     scanf("%d %d",&a,&b);
6     c=a%10;
7     d=b%10;
8     if(c==d)
9         printf("true\n");
10    else
11        printf("false\n");
12    return 0;
13 }
14 |
```

| | Input | Expected | Got | |
|---|-------|----------|-------|---|
| ✓ | 25 53 | false | false | ✓ |
| ✓ | 27 77 | true | true | ✓ |

Passed all tests! ✓

2. Intro to conditional statements

Problem statement:

Task

Given an integer, n , perform the following conditional actions:

- If n is odd, print **Weird**
- If n is even and in the inclusive range of **2** to **5**, print **Not Weird**
- If n is even and in the inclusive range of **6** to **20**, print **Weird**
- If n is even and greater than **20**, print **Not Weird**

Complete the stub code provided in your editor to print whether or not n is weird.

Input Format

A single line containing a positive integer, n .

Constraints

- $1 \leq n \leq 100$

Output Format

Print **Weird** if the number is weird; otherwise, print **Not Weird**.

Sample Output 0

Weird

Sample Input 1

24

Sample Output 1

Not Weird

Explanation

Sample Case 0: $n = 3$

n is odd and odd numbers are weird, so we print **Weird**.

Sample Case 1: $n = 24$

$n > 20$ and n is even, so it isn't weird. Thus, we print **Not Weird**.

Program:

```
1 #include<stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d",&n);
6     if((n%2==1)||((6<=n)&&(n<=20)))
7         printf("Weird");
8     else
9         printf("Not Weird");
10    return 0;
11 }
```

| | Input | Expected | Got | |
|---|-------|-----------|-----------|---|
| ✓ | 3 | Weird | Weird | ✓ |
| ✓ | 24 | Not Weird | Not Weird | ✓ |

Passed all tests! ✓

3. Pythagorean triangle:

Problem statement:

Three numbers form a Pythagorean triple if the sum of squares of two numbers is equal to the square of the third. For example, 3, 5 and 4 form a Pythagorean triple, since $3^2 + 4^2 = 25 = 5^2$. You are given three integers, a, b, and c. They need not be given in increasing order. If they form a Pythagorean triple, then print "yes", otherwise, print "no". Please note that the output message is in small letters. Sample Input 1 3 5 4 Sample Output 1 yes Sample Input 2 5 8 2 Sample Output 2 no

Program:

```
1 #include<stdio.h>
2 int main()
3 {
4     int a,b,c;
5     scanf("%d %d %d",&a,&b,&c);
6     if(a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
7         printf("yes");
8     else
9         printf("no");
10    return 0;
11
12 }
```

| | Input | Expected | Got | |
|---|-------------|----------|-----|---|
| ✓ | 3 5 4 | yes | yes | ✓ |
| ✓ | 5 8 2 | no | no | ✓ |

Passed all tests! ✓

4. Name that Shape:

Problem statement:

Write a program that determines the name of a shape from its number of sides. Read the number of sides from the user and then report the appropriate name as part of a meaningful message. Your program should support shapes with anywhere from 3 up to (and including) 10 sides. If a number of sides outside of this range is entered then your program should display an appropriate error message.

Sample Input 1

3

Sample Output 1

Triangle

Sample Input 2

7

Sample Output 2

Heptagon

Sample Input 3

11

Sample Output 3

The number of sides is not supported.

Program:

```
1 #include<stdio.h>
2 int main()
3 {
4     int n;
5     scanf("%d",&n);
6     switch (n)
7     {
8         case 3:
9             printf("Triangle");
10            break;
11        case 4:
12            printf("Quadrilaterals");
13            break;
14        case 5:
15            printf("Pentagon");
16            break;
17        case 6:
18            printf("Hexagon");
19            break;
20        case 7:
21            printf("Heptagon");
22            break;
23        case 8:
24            printf("Octagon");
25            break;
26        case 9:
27            printf("Nonagon");
28            break;
29        case 10:
30            printf("Decagon");
31            break;
32        default:
33            printf("The number of sides is not supported.");
34    }
35    return 0;
36 }
```

Test cases:

| | Input | Expected | Got | |
|---|-------|---------------------------------------|---------------------------------------|---|
| ✓ | 3 | Triangle | Triangle | ✓ |
| ✓ | 7 | Heptagon | Heptagon | ✓ |
| ✓ | 11 | The number of sides is not supported. | The number of sides is not supported. | ✓ |

Passed all tests! ✓

5. Chinese Zodiac:

Problem statement:

The Chinese zodiac assigns animals to years in a 12-year cycle. One 12-year cycle is shown in the table below. The pattern repeats from there, with 2012 being another year of the Dragon, and 1999 being another year of the Hare.

| Year | Animal |
|------|---------|
| 2000 | Dragon |
| 2001 | Snake |
| 2002 | Horse |
| 2003 | Sheep |
| 2004 | Monkey |
| 2005 | Rooster |
| 2006 | Dog |
| 2007 | Pig |
| 2008 | Rat |
| 2009 | Ox |
| 2010 | Tiger |
| 2011 | Hare |

Write a program that reads a year from the user and displays the animal associated with that year. Your program should work correctly for any year greater than or equal to zero, not just the ones listed in the table.

Sample Input 1

2004

Sample Output 1

Monkey

Sample Input 2

2010

Sample Output 2

Tiger

Program:

```
1 #include<stdio.h>
2 int main()
3 {
4     int y;
5     const char animals[][15]={ {"Dragon"}, {"Snake"}, {"Horse"}, {"Sheep"}, {"Monkey"}, {"Rooster"}, {"Dog"}, {"Pig"}}
6     scanf("%d",&y);
7     if(y>=0)
8     {
9         int i=y%12;
10        printf("%s",animals[i+4]);
11    }
12    else
13    {
14        printf("invalid year");
15    }
16    return 0;
17 }
```

Test cases:

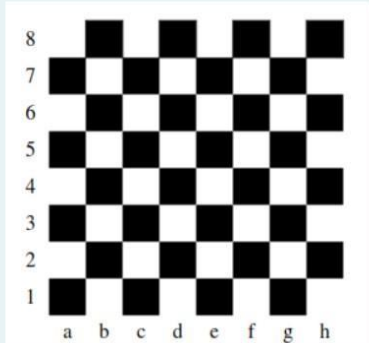
| | Input | Expected | Got | |
|---|-------|----------|--------|---|
| ✓ | 2004 | Monkey | Monkey | ✓ |
| ✓ | 2010 | Tiger | Tiger | ✓ |

Passed all tests! ✓

6. What color is that square?

Problem statement:

Positions on a chess board are identified by a letter and a number. The letter identifies the column, while the number identifies the row, as shown below:



Write a program that reads a position from the user. Use an if statement to determine if the column begins with a black square or a white square. Then use modular arithmetic to report the color of the square in that row. For example, if the user enters a1 then your program should report that the square is black. If the user enters d5 then your program should report that the square is white. Your program may assume that a valid position will always be entered. It does not need to perform any error checking.

Sample Input 1

a 1

Sample Output 1

The square is black.

Sample Input 2

d 5

Sample Output 2

The square is white.

Program:

```
1 #include<stdio.h>
2 int main()
3 {
4     int r;
5     char c;
6     scanf("%c %d",&c,&r);
7     if((c-'a')%2==0)
8         if(r%2==0)
9             printf("The square is white.");
10        else
11            printf("The square is black.");
12    else if(r%2==0)
13        printf("The square is black.");
14    else
15        printf("The square is white.");
16    return 0;
17 }
```

Test cases:

| | Input | Expected | Got | |
|---|-------|----------------------|----------------------|---|
| ✓ | a 1 | The square is black. | The square is black. | ✓ |
| ✓ | d 5 | The square is white. | The square is white. | ✓ |

Passed all tests! ✓

7. Suppandi & Area:

problem statement:

Suppandi is trying to take part in the local village math quiz. In the first round, he is asked about shapes and areas. Suppandi, is confused, he was never any good at math. And also, he is bad at remembering the names of shapes. Instead, you will be helping him [calculate the area](#) of shapes.

- When he says rectangle he is actually referring to a square.
- When he says square, he is actually referring to a triangle.
- When he says triangle he is referring to a rectangle
- And when he is confused, he just says something random. At this point, all you can do is say 0.

Help Suppandi by printing the correct answer in an integer.

Input Format

- Name of shape (always in upper case R à Rectangle, S à Square, T à Triangle)
- Length of 1 side
- Length of other side

Note: In case of triangle, you can consider the sides as height and length of base

Output Format

- Print the area of the shape.

Sample Input 1

T
10
20

Sample Output 1

200

Sample Input 2

S
30
40

Sample Output 2

600

Program:

```
1 #include<stdio.h>
2 int main()
3 {
4     int len,height,a=0;
5     char shape;
6     scanf("%c %d %d",&shape,&len,&height);
7     switch (shape)
8     {
9         case 83:
10             a=(len*height)/2;
11             break;
12         case 82:
13         case 84:
14             a=len*height;
15             break;
16         default:
17             a=0;
18     }
19     printf("%d",a);
20     return 0;
21 }
```

Test cases:

| | Input | Expected | Got | |
|---|---------------|----------|------|---|
| ✓ | T 10 20 | 200 | 200 | ✓ |
| ✓ | S 30 40 | 600 | 600 | ✓ |
| ✓ | B 2 11 | 0 | 0 | ✓ |
| ✓ | R 10 30 | 300 | 300 | ✓ |
| ✓ | S 40 50 | 1000 | 1000 | ✓ |

Passed all tests! ✓

8. Days of years

Problem statement:

Some data sets specify dates using the year and day of year rather than the year, month, and day of month. The day of year (DOY) is the sequential day number starting with day 1 on January 1st.

There are two calendars - one for normal years with 365 days, and one for leap years with 366 days. Leap years are divisible by 4. Centuries, like 1900, are not leap years unless they are divisible by 400. So, 2000 was a leap year.

To find the day of year number for a standard date, scan down the Jan column to find the day of month, then scan across to the appropriate month column and read the day of year number. Reverse the process to find the standard date for a given day of year.

Write a program to print the Day of Year of a given date, month and year.

Sample Input 1

18
6
2020

Sample Output 1

170

Program:

```
1 #include<stdio.h>
2 int main()
3 {
4     int d,m,y,a=0;
5     scanf("%d %d %d",&d,&m,&y);
6     for(int i=1;i<m;i++)
7     {
8         switch (i)
9         {
10             case 1:
11             case 3:
12             case 5:
13             case 7:
14             case 8:
15             case 10:
16             case 12:
17                 a+=31;
18                 break;
19             case 4:
20             case 6:
21             case 9:
22             case 11:
23                 a+=30;
24                 break;
25             case 2:
26                 if(((y%400==0)||((y%100!=0)&&(y%4==0)))
27                     a+=29;
28                 else
29                     a+=28;
30                 break;
31         }
32     }
33     a+=d;
34     printf("%d",a);
35     return 0;
36 }
```

Test cases:

| | Input | Expected | Got | |
|---|-----------------|----------|-----|---|
| ✓ | 18 6 2020 | 170 | 170 | ✓ |

Passed all tests! ✓

9. Superman's encounter

Problem statement:

Superman is planning a journey to his home planet. It is very important for him to know which day he arrives there. They don't follow the 7-day week like us. Instead, they follow a 10-day week with the following days: Day Number Name of Day 1 Sunday 2 Monday 3 Tuesday 4 Wednesday 5 Thursday 6 Friday 7 Saturday 8 Kryptonday 9 Coluday 10 Daxamday Here are the rules of the calendar: • The calendar starts with Sunday always. • It has only 296 days. After the 296th day, it goes back to Sunday. You begin your journey on a Sunday and will reach after n. You have to tell on which day you will arrive when you reach there.

Input format: •

Contain a number n ($0 < n$)

Output format: Print the name of the day you are arriving on

Example Input

7

Example Output

Kryptonday

Example Input

1

Example Output Monday

Program:

```
1 #include<stdio.h>
2 int main()
3 {
4     int n,d;
5     scanf("%d",&n);
6     d=(n%296)%10;
7     switch (d)
8     {
9         case 0:
10             printf("Sunday");
11             break;
12         case 1:
13             printf("Monday");
14             break;
15         case 2:
16             printf("Tuesday");
17             break;
18         case 3:
19             printf("Wednesday");
20             break;
21         case 4:
22             printf("Thursday");
23             break;
```

```

24     case 5:
25         printf("Friday");
26         break;
27     case 6:
28         printf("Saturday");
29         break;
30     case 7:
31         printf("Kryptonday");
32         break;
33     case 8:
34         printf("Coluday");
35         break;
36     case 9:
37         printf("Daxamday");
38         break;
39     default:
40         break;
41 }
42 return 0;
43 }

```

Test cases:

| | Input | Expected | Got | |
|---|-------|------------|------------|---|
| ✓ | 7 | Kryptonday | Kryptonday | ✓ |
| ✓ | 1 | Monday | Monday | ✓ |

Passed all tests! ✓