Exercise 3: Conditional and Alternative Statements

Nanda H Krishna

15 February 2018

1 Time conversion

Problem description: Write a program that asks the user for a time in 24-hour format, then displays it in 12-hour format. Represent the time with a pair of integers (hours, minutes).

Specification: The inputs are the hours and minutes with respect to 24-hour format and the output is the time in 12-hour format.

Program design: The program has one function `main()` where the conversion is carried out and the output is obtained.

Algorithm: The algorithm to is as follows:

```
int(input(hours, minutes))
f = 1
if hours >= 12:
  if hours is not 12 then hours -= 12
  f = 2
if hours == 0:
  hours = 12
print(hours, minutes, AM/PM)
```

Program:

```
#include<stdio.h>
int main()
{
  int h, m, f = 1;          // h - hours, m - minutes, f to check AM/PM
  scanf("%d %d", &h, &m);
  if(h >= 12) {
    if(h != 12) h -= 12;
    f = 2;                  // h > 12 implies PM, flag value 2
  }
  if(h == 0) h = 12;        // 12 AM
  printf("%d:%d", h, m);
  if(f == 1) printf(" AM"); // flag value 1
  else printf(" PM");       // flag value 2
  return 0;
}
```

Test Input:

```
21 36
```

Output:

```
9:36 PM
```

2 Time comparison

Problem description: Represent time by 3 integers representing the hours minutes, and seconds. Construct a function that takes two times and returns -1, 0, or +1 depending on whether the first time is earlier than the second one, same as the second one, or later than the second one.

Specification: The function `timecompare()` takes the two times as inputs and returns -1, 0 or 1 depending on whether the first time is lesser, equal to or greater than the second.

Prototype:

```
int timecompare(int t1[], int t2[])
```

Program design: The program consists of `timecompare(int t1[], int t2[])` which compares the times while `main()` receives input and tests the function.

Program:

```
#include<stdio.h>
int timecompare(int t1[], int t2[])
{
  int f;
  if(t1[0] > t2[0]) f = 1; // Hour comparison
  else if(t1[0] < t2[0]) f = -1;
  else {
    if(t1[1] > t2[1]) f = 1; // Minute comparison
    else if(t1[1] < t2[1]) f = -1;
    else {
      if(t1[2] > t2[2]) f = 1; // Second comparison
      else if(t1[2] < t2[2]) f = -1;
      else f = 0; // Equal case
    }
  }
  return f;
}
int main()
{
  int t1[3], t2[3], td;
  scanf("%d%d%d%d%d%d",&t1[0],&t1[1],&t1[2],&t2[0],&t2[1],&t2[2]);
  // Input in 24H format for t1 and t2 as H M S
  td = timecompare(t1,t2);
  if(td == -1) printf("Time 1 is lesser than Time 2");
  else if(td == 1) printf("Time 1 is greater than Time 2");
  else printf("Time 1 and Time 2 are equal");
  return 0;
}
```

Test Input:

```
13 22 29 8 45 32
```

Output:

```
Time 1 is greater than Time 2
```

3 Time difference

Problem description: Write a program to calculate the time difference between two times, represented in (hours, minutes, seconds) format.

Specification: The function `timedif()` takes the two times as inputs and outputs on `stdout` the difference between them.

Prototype:

```
void timedif(int t1[], int t2[])
```

Program design: The program consists of `timecompare(int t1[], int t2[])` to compare which time is larger, `timedif(int t1[], int t2[])` to compute the difference between the times and `main()` to receive inputs and test the functions.

Program:

```
#include<stdio.h>
int timecompare(int t1[], int t2[])
```

```
{
  int f;
  if(t1[0] > t2[0]) f = 1; // Hour comparison
  else if(t1[0] < t2[0]) f = -1;
  else {
    if(t1[1] > t2[1]) f = 1; // Minute comparison
    else if(t1[1] < t2[1]) f = -1;
    else {
      if(t1[2] > t2[2]) f = 1; // Second comparison
      else if(t1[2] < t2[2]) f = -1;
      else f = 0; // Equal case
    }
  }
  return f;
}
void timedif(int t1[], int t2[])
{
  int dc = timecompare(t1,t2); // Check greater time for -ve sign avoid
  int dif[3] = {0, 0, 0};
  dif[2] = t1[2]*dc - t2[2]*dc;
  if(dif[2] < 0) {
    dif[1] = -1;
    dif[2] += 60;
  }
  dif[1] += (t1[1]*dc - t2[1]*dc);
  if(dif[1] < 0) {
    dif[0] = -1;
    dif[1] += 60;
  }
  dif[0] += (t1[0]*dc - t2[0]*dc);
  printf("%d:%d:%d", dif[0], dif[1], dif[2]);
}
int main()
{
  int t1[3], t2[3], td;
  scanf("%d%d%d%d%d%d",&t1[0],&t1[1],&t1[2],&t2[0],&t2[1],&t2[2]);
  // Input in 24H format for t1 and t2 as H M S
  timedif(t1,t2);
  return 0;
}
```

Test Input:

```
12 20 24 0 45 59
```

Output:

```
11:34:25
```

4 Income Tax

Problem description: How much tax you should pay depends upon the tax slab applicable to your income.

1. Income Tax Slab for Individual Tax Payers (Less Than 60 Years Old)

Income Slab	Tax Rate
Up to Rs.2,50,000	No tax
Rs.2,50,000 - Rs.5,00,000	5%
Rs.5,00,000 - Rs.10,00,000	20%
Rs.10,00,000 and beyond	30%

To the tax, add cess: 3% on total of income tax.

2. Income Tax Slab for Senior Citizens (60 Years Old Or More but Less than 80 Years Old)

Income Slab	Tax Rate
Up to Rs.3,00,000	No tax
Rs.3,00,000 - Rs.5,00,000	5%
Rs.5,00,000 - Rs.10,00,000	20%
Rs.10,00,000 and beyond	30%

3. Income Tax Slab for Senior Citizens (More than 80 Years Old)

Income Slab	Tax Rate
Up to Rs.2,50,000	No tax
Rs.2,50,000 - Rs.5,00,000	No tax
Rs.5,00,000 - Rs.10,00,000	20%
Rs.10,00,000 and beyond	30%

Specification: The inputs to the function `tax()` are the salary and the age while the output is the tax to be paid.

Prototype:

```
float tax(float i, int a)
```

Program design: The program consists of `tax(float i, int a)` and `main()` alone. The `tax()` function returns the tax to be paid while `main()` calls the function for testing.

Program:

```
#include<stdio.h>
float tax(float i, int a)
{ float tax = 0;

  if(a < 60) {
    if(i < 250000) tax = 0;
    else if(i < 500000) tax = (5/100.0)*(i-250000);
    else if(i < 1000000) tax = (5/100.0)*(250000) + (20/100.0)*(i-500000);
    else tax = (5/100.0)*(250000) + (20/100.0)*(500000) + (30/100.0)*(i-1000000)
    tax *= (103/100.0);
  }
  else if(a < 80) {
    if(i < 300000) tax = 0;
    else if(i < 500000) tax = (5/100.0)*(i-300000);
    else if(i < 1000000) tax = (5/100.0)*(200000) + (20/100.0)*(i-500000);
    else tax = (5/100.0)*(200000) + (20/100.0)*(500000) + (30/100.0)*(i-1000000)
  }
  else {
    if(i < 500000) tax = 0;
    else if(i < 1000000) tax = (20/100.0)*(i-500000);
    else tax = (20/100.0)*(500000) + (30/100.0)*(i-1000000);
  }
  return tax;
}
int main(void)
{
  int income, age;
  scanf("%d %d", &income, &age);
  printf("%f", tax(income, age));
  return 0;
}
```

Test Input:

```
500000 60
```

Output:

```
10000.0
```

5 Electricity fee

Problem description: Construct a tariff calculator for the Domestic Electricity Bills of TNEB, based on the following slab rates:

1. Consumption upto 100 units: free.
2. Consumption above 100 units and upto 200 units: Rs 1.50 per unit.
3. Consumption above 200 units and upto 500 units: Rs 2.00 per unit for 101-200 units and Rs 3.00 per unit for 201-500 units.
4. Consumption above 500 units: Rs 3.50 per unit for 101-200 units, Rs 4.60 per unit for 201-500 units, and Rs 6.60 beyond 500 units.

Specification: The function `fee()` takes the number of units as parameter and the output is the amount to be paid as electricity fee.

Prototype:

```
float fee(int units)
```

Program design: The program consists of `fee(int units)` which calculates the tariff given the units while `main()` reads input and tests the function call.

Program:

```
#include<stdio.h>
float fee(int units)
{
  float fees = 0;
  if(units <= 100) {
    fees = 0;
  }
  else if(units <= 200) {
    fees = 1.50 * (units - 100);
  }
  else if(units <= 500) {
    fees = 2.00 * 100 + 3.00 * (units - 200);
  }
  else {
    fees = 3.50 * 100 + 4.60 * 300 + 6.60 * (units - 500);
  }
  return fees;
}
int main()
{
  int u;
  scanf("%d", &u); // Input units
  printf("%.2f", fee(u));
  return 0;
}
```

Test Input:

```
670
```

Output:

```
2852.0
```

6 Grading

Problem description: Write a function to translate the marks of a student in a semester into letter grades. Assume 8 exams in a semester. Let your program read 8 marks, then print the marks and the grades.

Mark range	Grade points	Leter grade
91-100	10	S
81-90	9	A
71-80	8	B
61-70	7	C
57-60	6	D
51-56	5	E
<50	0	U

Specification: The function `grade()` takes an array of marks as input and assigns a grade to each mark. These are printed on `stdout`.

Prototype:

```
void grade(int marks[])
```

Program design: The program consists of `grade(int marks[])` which performs the grading while `main()` gets the inputs and tests the function.

Program:

```
#include<stdio.h>
void grade(int marks[])
{
  char grades[8];
  for(int i = 0; i < 8; i++) {
    if(marks[i] >= 91 && marks[i] <= 100) {
      grades[i] = ’S’;
    }
    else if(marks[i] >= 81 && marks[i] <= 90) {
      grades[i] = ’A’;
    }
    else if(marks[i] >= 71 && marks[i] <= 80) {
      grades[i] = ’B’;
    }
    else if(marks[i] >= 61 && marks[i] <= 70) {
      grades[i] = ’C’;
    }
    else if(marks[i] >= 57 && marks[i] <= 60) {
      grades[i] = ’D’;
    }
    else if(marks[i] >= 51 && marks[i] <= 56) {
      grades[i] = ’E’;
    }
    else {
      grades[i] = ’U’;
    }
  }
  for(int p = 0; p < 8; p++) {
    printf("%d %c\n", marks[p], grades[p]);
  }
}
int main()
{
  int marks[8];
  for(int i = 0; i < 8; i++) {
    scanf("%d", &marks[i]);
  }
  grade(marks);
  return 0;
}
```

Test Input:

```
100 88 72 64 57 54 50 23
```

Output:

```
100  S
 88  A
 72  B
 64  C
 57  D
 54  E
 50  U
 23  U
```

7 Maximum and Minimum

Problem description: Write a program that finds the smallest and the largest of four integers entered by the user.

Specification: The function `main()` takes user's input and calculates the minimum and maximum values, and displays them on `stdout`.

Program design: The inputs are received in `main()`, where the computation is also done for minimum and maximum. Output is displayed on `stdout`.

Program:

```
#include<stdio.h>
int main()
{
  int max, min, t;
  for(int i = 0; i < 4; i++) {
    scanf("%d", &t);
    if(i == 0) {
      max = t; // First number is made max
      min = t; // First number is made min
    }
    else {
      if(max < t) max = t; // Compare and change max if required
      if(min > t) min = t; // Compare and change min if required
    }
  }
  printf("%d %d", max, min);
  return 0;
}
```

Test Input:

```
10 20 32 2
```

Output:

```
32 2
```

8 Inversions

Problem description: In a sequence of integers `a0, a1, a2, a3`, any pair of integers `(ai, aj)` is said to be an inversion if `ai > aj` for `i < j`. Write a program to correct/order all the inversions in the sequence.

Specification: The function `main()` takes user's input and reorders the numbers to remove inversions, and displays them on `stdout`.

Program design: The inputs are received in `main()`, where the removal of inversions is also performed. Output is displayed on `stdout`.

Program:

```
#include <stdio.h>
#include <stdbool.h>
int main ()
{
  int a, b, c, d;
  int t;
  bool inverted;
  int n;
  scanf("%d%d%d%d", &a, &b, &c, &d);
  inverted = true;
  n = 0;
  while (inverted) {
    inverted = false;
    if (a > b) {
      t = a; a = b; b = t;
      inverted = true;
      n += 1;
    }
    else if (b > c) {
      t = b; b = c; c = t;
      inverted = true;
      n += 1;
    }
    else if (c > d) {
      t = c; c = d; d = t;
      inverted = true;
      n += 1;
    }
  }
  printf ("%d %d %d %d\n%d", a, b, c, d, n);
  return 0;
}
```

Test Input:

```
40 30 20 10
```

Output:

```
10 20 30 40
6
```