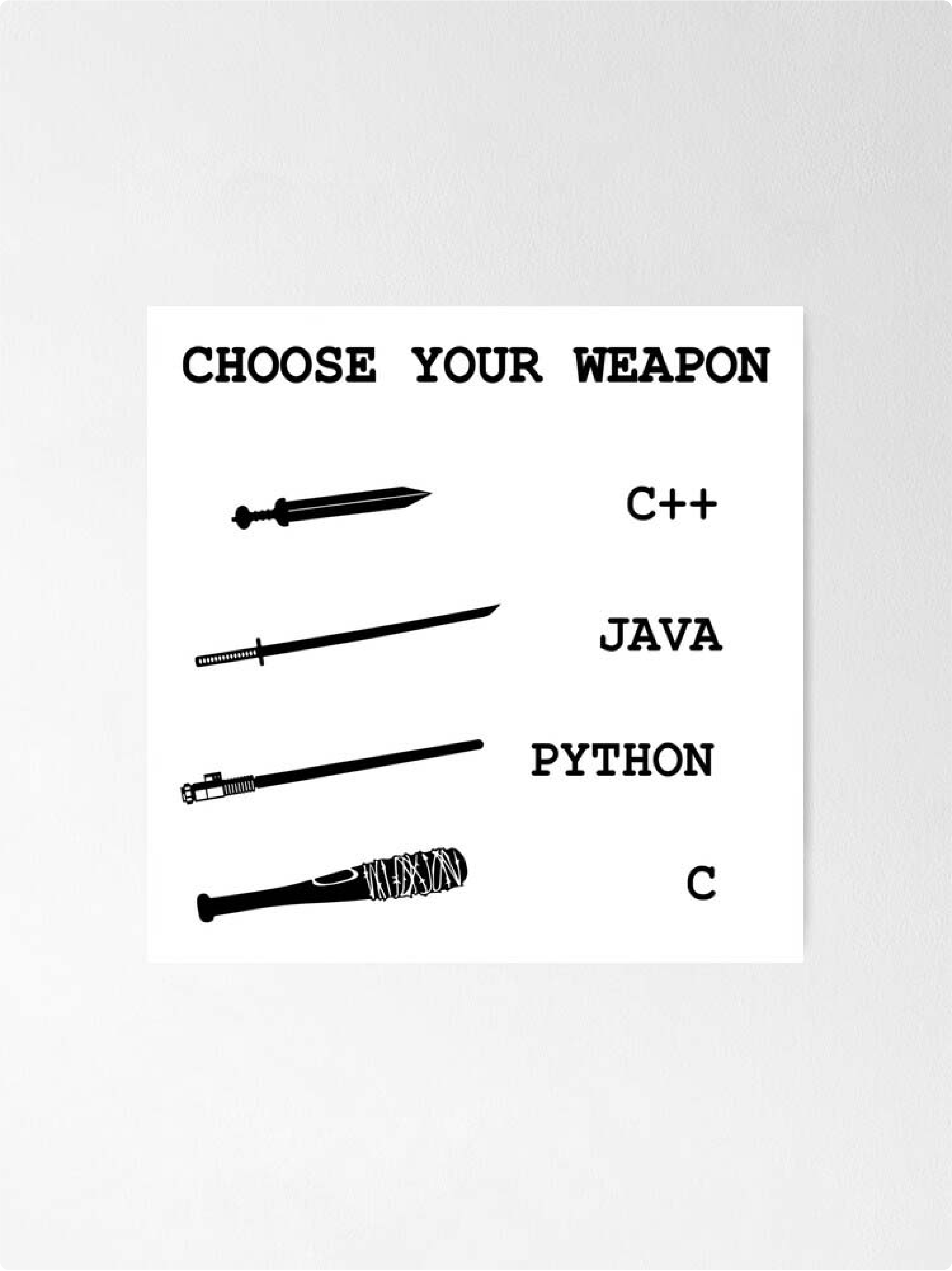

Unit 2.1: Logic conditions and if-else statements

Using the if statement

Problem 1:

You are planning a car trip so you post on a carpooling website in order to share the cost of the trip.

If you have 0 passengers the carpool site does not charge anything and you alone pay the full cost of the trip. If you have 1 or more passengers the carpool site adds a $1 fee to the cost of the trip and evenly divides the total cost ($1 fee + gas) among the passengers and you. You want to write a program that calculates the cost you have to pay. The program should read the number of passengers (an integer) and the cost of gas for the trip (a decimal number). The program should then print the cost that you have to pay (a decimal number) with 2 digits after the decimal point.

Examples

Input

```
0 23.9
```

Output

```
23.90
```

```c
#include <stdio.h>

int main(void){
    int man;
    double gas;
    scanf("%d %lf", &man, &gas);

    double cost;
    if(man==0){
        cost = gas;
    }
    else{
        cost = (gas + 1.0)/(man+1);
    }
    printf("%.2lf", cost);
    return 0;
}
```

Probelm 2:

The hostel in which you plan to spend the night tonight offers very interesting rates, as long as you do not arrive too late. Housekeeping finishes preparing rooms by noon, and the sooner guests arrive after noon, the less they have to pay. You are trying to build a C program that calculates your price to pay based on your arrival time.

Your program will read an integer (between 0 and 12) indicating the number of hours past noon of your arrival. For example, 0 indicates a noon arrival, 1 a 1pm arrival, 12 a midnight arrival, etc. The base price is 10 dollars, and 5 dollars are added for every hour after noon. Thankfully the total is capped at 53 dollars, so you'll never have to pay more than that. Your program should print the price (an integer) you have to pay, given the input arrival time.

Example

Input

7

Output

45

```c
#include <stdio.h>

int main(void){
    int pastNoon; // 0 = noon; 12 = midnight ++5/h
    int max = 53;

    int cost;
    scanf("%d", &pastNoon);
    cost = 10 + (pastNoon * 5);
    int cap = (cost>max);
    if(cap){
        cost = max;
    }

    printf("%d", cost);
    return 0;
}
```

Comparing Decimal Number

Problem 1:

You arrive in front of a bridge that you must cross to reach a village before dark. Crossing the bridge is not free - the bridgekeeper asks you to roll two dice to determine the cost. You decide to write a program to verify that he is charging the right price.

Your program should read two integers, between 1 and 6, representing the values of each die. If the sum is greater than or equal to 10, then you must pay a special fee (36 coins). Otherwise, you pay twice the sum of the values of the two dice. Your program must then display the text "Special tax" or "Regular tax" followed by the amount you have to pay on the next line.

Example

Input

```
5
6
```

Output

```
Special tax
36
```

```c
#include <stdio.h>

int main(void){
    int x, y;
    int price;
    scanf("%d", &x);
    scanf("%d", &y);
    price = x+y;
    if(price >= 10){
        printf("Special Tax\n%d", 36);
    }else{
        printf("Regular Tax\n%d", price*2);
    }
    return 0;
}
```

Probelm 2:

You decide to bet on the final match of the Tug of War National Championship.

Prior to the match the names and weights of the players are presented, alternating by team (team 1 player 1, team 2 player 1, team 1 player 2, and so on). There is the same number of players on each side. You record the player weights as they are presented and calculate a total weight for each team to inform your bet. You write a C program to assist with this.

Your program should first read an integer indicating the number of members per team. Then, the program should read the player weights (integers representing kilograms) alternating by team.

After calculating the total weight of each team, the program should display the text

"Team X has an advantage" (replacing X with 1 or 2 depending on which team has a greater total weight).

You will then display the text "Total weight for team 1:" followed by the weight of team 1, then "Total weight for team 2:" followed by the weight of team 2 (see example below).

You are guaranteed that the two teams will not have the same total weight.

Example

Each team is composed of four players. Those of the first weigh 110, 113, 112, and 117kg, while those of the second weigh 106, 102, 121, and 111kg. Team 1 weighs a total of 452kg whereas team 2 weighs a total of 440kg, giving team 1 an advantage.

```
Input:
4
110
106
113
102
112
121
117
111
```

```
Output:
Team 1 has an advantage
Total weight for team 1: 452
Total weight for team 2: 440
```

```
#include <stdio.h>

int main(void){
    int mem, tim1, tim2, temp;
    scanf("%d", &mem);

    tim1 = 0;
    tim2 = 0;
    int i, j = 0;
    for(i=0;i<mem;i++){
        for(j=0;j<2;j++){
            scanf("%d", &temp);
            if(j==0){
                tim1 = tim1 + temp;
            } else{
                tim2 = tim2 + temp;
            }
        }
    }

    if(tim1>tim2){
        printf("Team 1 has an advantage\n");
    }else{
        printf("Team 2 has an advantage\n");
    }
    printf("Total weight for team 1: %d\nTotal weight for team 2: %d", tim1, tim2)

    return 0;
}
```

Combining logic conditions using the logical AND and OR

Problem 1

The hostel in which you stop for the night changes its prices according to the age of the customer and the weight of their luggage. The rules are not very clear, so you decide to write a small program that will easily allow you and your travel companions to know the price of one night.

One room costs nothing if you are exactly 60 (the age of the innkeeper), or 5 dollars if you are less than 10 years old. For everyone else, the cost is 30 dollars plus an additional 10 dollars if you bring more than 20 pounds of luggage. Your program should read the customer's age first, then the weight of their luggage, then output the price they have to pay.

Example

Input:

22

25

Output:

40

```c
#include <stdio.h>

int main(void){
    int age, lb, price;
    scanf("%d %d", &age, &lb);

    if(age == 60){
        price = 0;
    }else{
        if(age < 10){
            price = 5;
        }else{
            if(age>=10 && age != 60){
                price = 30;
                if(lb>10){
                    price = price + 10;
                }
            }
        }
    }
    printf("%d", price);
    return 0;
}
```

Negating a logic condition using the logic operator NOT

Problem 1

As you cross a forest you can't help but admire the nature around you including the many species of trees. Despite your interest, you are a very unskilled botanist and have a lot of trouble identifying different trees. A friend gives you some guidance and you decide to write a program that will give you the name of the tree based on its characteristics.

There are 4 types of trees:

the "Tinuviel" is 5 meters high or less and its leaves are composed of 8 or more leaflets the "Calaelen" is 10 meters high or more and its leaves are composed of 10 or more leaflets

the "Falarion" is 8 meters high or less and its leaves are composed of 5 or fewer leaflets

the "Dorthonion" is 12 meters tall or more and its leaves are composed of 7 or fewer leaflets

Your program should read the height and the number of leaflets of a given tree (both integers), and should be able to determine and display the name of the corresponding tree. If the height and number of leaflets does not match any of the tree type descriptions, your program should display "Uncertain".

Example

Input

```
12
12
```

Output

```
Calaelen
```

```c
#include <stdio.h>
int main() {
    int height = 0;
    int nbLeaflets = 0;
    scanf("%d %d", &height, &nbLeaflets);
    int tinuviel = height <= 5 && nbLeaflets >= 8;
    int calaelen = height >= 10 && nbLeaflets >= 10;
    int falarion = height <= 8 && nbLeaflets <= 5;
    int dorthonion = height >= 12 && nbLeaflets <= 7;
    if(tinuviel) {
        printf("Tinuviel");
    }
    if(calaelen){
        printf("Calaelen");
    }
    if(falarion){
        printf("Falarion");
    }
    if(dorthonion){
        printf("Dorthonion");
    }
    if(!tinuviel && !calaelen && !falarion && !dorthonion){
        printf("Uncertain");
    }
    return 0;
}
```

Unit 2.2: Arrays, for and while loops

Using arrays of integers

Problem 1:

Your grandparents gave you a fantastic cooking recipe but you can never remember how much of each ingredient you have to use! There are 10 ingredients in the recipe and the quantities needed for each of them are given as input (in grams). Your program must read 10 integers (the quantities needed for each of the ingredients, in order) and store them in an array. It should then read an integer which represents an ingredient's ID number (between 0 and 9), and output the corresponding quantity.

Example

Input:

500 180 650 25 666 42 421 1 370 211

3

Output:

25

```c
#include <stdio.h>

int main(void){
    int ings[10];
    int i;

    for(i=0;i<10;i++){
        scanf("%d ", &ings[i]);
    }

    int check;
    scanf("%d", &check);
    printf("%d", ings[check]);

    return 0;
}
```

Repeating instructions with a FOR loop

Problem 1:

You are responsible for a rail convoy of goods consisting of several boxcars. You start the train and after a few minutes you realize that some boxcars are overloaded and weigh too heavily on the rails while others are dangerously light. So you decide to stop the train and spread the weight more evenly so that all the boxcars have exactly the same weight (without changing the total weight). For that you write a program which helps you in the distribution of the weight.

Your program should first read the number of cars to be weighed (integer) followed by the weights of the cars (doubles). Then your program should calculate and display how much weight to add or subtract from each car such that every car has the same weight. The total weight of all of the cars should not change. These additions and subtractions of weights should be displayed with one decimal place.

You may assume that there are no more than 50 boxcars.

Example 1

In this example, there are 5 boxcars with different weights summing to 110.0. The ouput shows that we are modifying all the boxcars so that they each carry a weight of 22.0 (which makes a total of 110.0 for the entire train). So we remove 18.0 for the first boxcar, we add 10.0 for the second, we add 2.0 for the third, etc.

Example

Input

```
5
40.0
12.0
20.0
5.0
33.0
```

Output

```
-18.0
10.0
2.0
17.0
-11.0
```

```c
#include <stdio.h>

int main(void){
    int i, cars = 0;
    scanf("%d", &cars);

    double perW, temp, agg = 0;
    double per[cars];

    double total = 0;
    for(i=0;i<cars;i++){
        scanf("%lf", &temp);
        total = total + temp;
        per[i] =  temp;
    }

    perW = total/(double) cars;

    for(i=0;i<cars;i++){
        agg = perW - per[i];
        printf("%.1lf\n", agg);
    }

    return 0;
}
```

Using an IF statement inside a FOR loop

Problem 1:

You plan to make a delicious meal and want to take the money you need to buy the ingredients. Fortunately you know in advance the price per pound of each ingredient as well as the exact amount you need. The program should read in the number of ingredients (up to a maximum of 10 ingredients), then for each ingredient the price per pound. Finally your program should read the weight necessary for the recipe (for each ingredient in the same order). Your program should calculate the total cost of these purchases, then display it with 6 decimal places.

Example

There are 4 ingredients and they all have a different price per pound: 9.90, 5.50, 12.0, and 15.0. You must take 0.25 lbs of the first, 1.5 lbs of the second, 0.3 lbs of the third and 1 lb of the fourth. It will cost exactly $29.325000.

Input:

```
4
9.90 5.50 12.0 15.0
0.250 1.5 0.300 1.0
```

Output:

```
29.325000
```

```c
#include <stdio.h>

int main(void){
    int ings = 0;
    scanf("%d", &ings);

    double pcs[ings];
    double lbs[ings];

    int i;
    double total = 0.0;

    for(i = 0; i<ings; i++){
        scanf("%lf ", &pcs[i]);
    }

    for(i=0; i<ings;i++){
        scanf("%lf", &lbs[i]);
        total = total + (pcs[i] * lbs[i]);
    }

    printf("%.6lf", total);
    return 0;
}
```

Nesting IF and FOR

Probelm 1:

You want to determine the number of cities in a given region that have a population strictly greater than 10,000. To do this, you write a program that first reads the number of cities in a region as an integer, and then the populations for each city one by one (also integers).

Example

Input

```
6
1000
5000
15000
4780
0
23590
```

Output

```
2
```

```c
#include <stdio.h>

int main(void){
    int cts, i, pop = 0;
    int ctr = 0;
    scanf("%d", &cts);

    for(i=0;i<cts;i++){
        scanf("%d", &pop);
        if(pop>10000){
            ctr++;
        }
    }
    printf("%d", ctr);
    return 0;
}
```

Repeating inside repetition

Problem 1:

Create a program that displays on the screen a square of n x n *, with the integer n given as an input.

Examples

Input:

5

Output:

```
*****

*****

*****

*****

*****
```

```c
#include <stdio.h>

int main(void){
    int n, i, j;
    scanf("%d", &n);

    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Repeating using a WHILE loop

Problem 1:

Much of the work of a university administration (in addition to managing teachers, researchers, students, courses, etc.) is to ensure the proper functioning of the university and in particular that the accounting job is well done. Once a year, an annual report of expenditures must be made.

All expenses for the year have been recorded and classified in a multitude of files and the sum of all these expenses must now be calculated. But no one knows exactly how many different expenses have been made in the past year!

Your program will have to read a sequence of positive integers and display their sum. We do not know how many integers there will be, but the sequence always ends with the value -1 (which is not an expense, just an end marker).

Example

Input

```
1000
2000
500
-1
```

Output

```
3500
```

```
#include <stdio.h>

int main(void){
    int tot= 0;
    int temp = 0;
    int marker = -1;

    while(temp != marker){
        scanf("%d", &temp);
        if(temp == marker){
            break;
        }else{
            tot = tot + temp;
        }

    }

    printf("%d", tot);
    return 0;
}
```

Efficiently using a WHILE loop

Problem 1:

In order to be able to better fight various epidemics in a region, the department of medicine of a major university has launched a large study. Researchers are interested in how fast an epidemic spreads, and therefore the speed at which health measures must be put in place. Your program should first read an integer representing the total population of the area. Knowing that a person was infected on day 1 and that each infected person contaminates two new people every day, you must calculate the day at which the entire population of the area will be infected.

> For a total population of 100 inhabitants, on day 1 a single person is infected. This is followed by 2 new people on the second day for a total of 3 infected people. On the third day, 6 new people are infected for a total of 9 infected people. On the fourth day, 18 new people are infected, for a total of 27 people. On the fifth day, 27 new people infect 54 people for a total of 81 people infected. On the sixth day, the last of the 100 people will be infected (though the 81 people have the potential to contaminate 162 people) so your program shuld output '6'.
>
> > Input:
> >
> > 100
>
> > Output:
> >
> > 6

```c
#include <stdio.h>

int main(void){
    int pops = 0;
    int infected = 1;
    int days = 1;
    scanf("%d", &pops);

    while(infected<pops){
        infected = infected*3;
        days++;
    }
    printf("%d", days);
    return 0;
}
```

Practicing WHILE loops

Problem 1:

We would like you to develop a program capable of making a child play automatically the game of "more or less" – the child must try to guess a secret number. The program should respond to each guess with "it is more" or "it is less" until the child finds the right number.

Your program must first read an integer indicating the number that the child will have to find during the game. Next the program should repeatedly read the player's guesses and display the text "it is more" if the child has submitted a smaller number or "it is less" if they have submitted a larger number. Once the correct number is reached, the program should print "Number of tries needed:" followed by a new line and the integer number of tries that it took the guesser.

Example

Input

```
5
1 2 3 4 5
```

Output

```
it is more
it is more
it is more
it is more
Number of tries needed:
5
```

```c
#include <stdio.h>

int main(void){
    int target, guess;
    int ctr = 1;
    scanf("%d", &target);

    while(guess != target){
        scanf("%d", &guess);
        if(guess == target){
            break;
        }
        if(guess<target){
            printf("It is more\n");
        }else{
            printf("It is less\n");
        }
        ctr++;
    }
    printf("Number of tries needed:\n%d", ctr);
    return 0;
}
```

Problem 2:

University chemists have developed a new process for the manufacturing of a drug that heals wounds extremely quickly. The manufacturing process is very lengthy and requires monitoring the chemicals at all times, sometimes for hours! Entrusting this task to a student is not possible; students tend to fall asleep or not pay close attention after a while. Therefore you need to program an automatic device to monitor the manufacturing of the drug. The device measures the temperature every 15 seconds and provides these measurement to your program.

Your program should first read two integers representing the minimum and maximum safe temperatures. Next, your program should continuously read temperatures (integers) that are being provided by the device. Once the chemical reaction is complete the device will send a value of -999, indicating to you that temperature readins are done. For each recorded temperature that is in the correct range (it could also be equal to the min or max values), your program should display the text "Nothing to report". But as soon as a temperature reaches an unsafe level your program must display the text "Alert!" and stop reading temperatures (although the device may continue sending temperature values).

Examples

Input:

```
10 20
15 10 20 0 15 -999
```

Output:

```
Nothing to report
Nothing to report
Nothing to report
Alert!
```

```
#include <stdio.h>

int main(void){
    int min, max, temp;
    int fin = -999;
    scanf("%d %d", &min, &max);
    scanf("%d", &temp);

    while(temp != fin){
        // scanf("%d", &temp);
        // if(temp == fin){
        //     break;
        // }
        if(temp >= min && temp<= max){
            printf("Nothing to report\n");
        }else{
            printf("Alert!\n");
            break;
        }
        scanf("%d", &temp);
    }

    return 0;
}
```

Unit 2.3: Strings, sort and search algorithms

Using strings (arrays of characters)

Problem 1:

Write a C-program that prints out a word as many times as specified. The number of repetitions and the word should be given as input to the program. You may assume that the word has no more than 100 characters (be sure to also reserve space for the null terminator, \0, though!).

Examples

Input:

2 Hello

Output:

Hello

Hello

```c
#include <stdio.h>

int main(void){
    char words[101];
    int reps;
    scanf("%d %s", &reps, words);

    int i;
    for(i=0;i<reps;i++){
        printf("%s\n", words);
    }
    return 0;
}
```

Using the special null terminator (\0) to identify the end of a string

Problem 1:

Your local public library keeps a record of all of its patrons, consisting of index cards that hold a person's last name followed by their first name (so that the cards can easily be sorted alphabetically by last name). Unfortunately a computer error led to incorrectly printed forms last month, resulting in a number of cards that list the patron's first name followed by their last name rather than the other way around. Your job is to read these pairs of first and last names and display them in the correct order (last name followed by first name). You may assume that each first and last name has at most 100 characters and does not contain any spaces.

Your program should first read the total number of names (an integer) in order to know how many index cards need to be processed altogether. Next, for each index card, your program should read a patron's first name and last name and then display these names correctly, that is on one line, the last name followed by one space, followed by the first name. Your program should print the reversed name immediately after reading the patron's names (ie, it should not wait until it has read all of the index cards to begin printing).

Note that, for ease of viewing, the example below shows all of the inputs in one block and all of the outputs in another block, despite the fact that programmatically these will be interspersed.

Example

Input:

```
4
Alan Turing
Ada Lovelace
Donald Knuth
Claude Shannon
```

Output:

```
Turing Alan
Lovelace Ada
Knuth Donald
Shannon Claude
```

```c
#include <stdio.h>

int main(void){
    int num;
    char aName[101];
    char iName[101];

    scanf("%d", &num);

    int i;
    for(i = 0; i<num;i++){
        scanf("%s %s", iName, aName);
        printf("%s %s\n", aName, iName);
    }
    return 0;
}
```

Finding the length of a string

Problem 1:

At the annual meeting of MOOC fans, participants register on the first day of the event in order to receive their name tags, brochures and banquet vouchers. Unfortunately this often results in long lines. In an attempt to speed things up, there are now two people working the registration desk: one person who has the registration materials for those fans whose names contain an odd number of letters, the other for those whose names have an even number of letters. Your job is to write a C-program that will tell a fan which line to stand in.

To simplify the program, you may assume that student names are less than 50 characters long and contain no spaces. Your program should output an integer value (1 or 2) depending on whether the fan should join line 1 (name has even number of letters) or line 2 (name has odd number of letters).

Examples

Input:
Sharrock

Output:
1

```c
#include <stdio.h>

int main(void){
    char name[50];
    int ctr = 0;

    scanf("%s", name);
    while(name[ctr] != '\0'){
        ctr++;
    }

    if(ctr%2 == 0){
        printf("%d", 1);
    }else{
        printf("%d", 2);
    }
    return 0;
}
```

Working with string lengths

Problem 1:
Your job is to find the length of the longest word in a text with no punctuation or special characters of any kind - only contains words. To do so, please write a C-program that takes as a input first the number of words in a text, followed by all of the words in the text. The output of your program should be the length of the longest word in the text.

To simplify your program, you can assume that the longest word will not exceed 100 characters.

Examples

Input:

14

This is a simple example text

we have to find the largest word length

Output:

7

```c
#include <stdio.h>

int main(void){
    int words;
    int longest = 0;
    int i = 0;
    int ctr = 0;
    char text[101];
    scanf("%d", &words);

    for(i=0;i<words;i++){
        scanf("%s", text);
        while(text[ctr] != '\0'){
          ctr++;
        }
        // printf("%d\n", ctr);
        if(ctr>longest){
          longest = ctr;
        }
        ctr = 0;
    }

    printf("%d", longest);

return 0;
}
```

Sorting strings

Problem 1:

You are conducting a linguistic study and are interested in finding words that contain the letter 't' or 'T' in the first half of the word (including the middle letter if there is one). Specifically, if the first half of the word does contain a 't' or a 'T', your program should output a 1. If the first half does not contain the letter 't' or 'T', but the second half does, then your program should output a 2. Otherwise, if there is no 't' or 'T' in the word at all, your program's output should be -1. You may assume that the word entered does not have more than 50 letters.

Examples

Input:

```
apple
```

Output:

```
-1
```

Input:

```
raincoat
```

Output:

```
2
```

Input:

```
enter
```

Output:

```
1
```

```c
#include <stdio.h>

int main(void){
    char word[51];
    scanf("%s", word);

    int ctr = 0;
    int found = 0;
    int log = 0;
    while(word[ctr] != '\0'){
        if(word[ctr] == 't' || word[ctr] == 'T'){
            found = 1;
            log = ctr;
        }
        ctr++;
    }

    if(log>=((ctr/2)+1)){
        printf("%d", 2);
    }else{
        if(!found){
            printf("%d", -1);
        }else{
            printf("%d", 1);
        }
    }

    return 0;
}
```

Searching with bisection and sorting with bubble sort

Problem 1:

You are still conducting linguistic research! This time, you'd like to write a program to find out how many letters occur multiple times in a given word. Your program should read a word from the input and then sort the letters of the word alphabetically (by their ASCII codes). Next, your program should iterate through the letters of the word and compare each letter with the one following it. If these equal each other, you increase a counter by 1, making sure to then skip ahead far enough so that letters that occur more than twice are not counted again. You may assume that the word you read from the input has no more than 50 letters, and that the word is all lowercase.

Example

Input:

erroneousnesses

Output:

5

```c
#include <stdio.h>

int main(void){
  char word[51];
 scanf("%s", word);

 int i = 0;
 int j = 0;
 int n = 0;
 char swap;
 int ctr = 0;
 int act = 0;

 while(word[n]!='\0'){
   n++;
 }

 for(j = 0; j<n-1;j++){
   for(i = 0; i<n-1; i++){
    if(word[i]>word[i+1]){
       swap = word[i+1];
       word[i+1] = word[i];
       word[i] = swap;
     }
   }
 }

 //printf("%s\n", word);

 i = 0;
 while(word[i] != '\0'){
   if(word[i] == word[i+1]){
     act++;
   }else{
    if(act>=1){
     ctr++;
    }
    act = 0;
   }

   //printf("%c => %d\n", word[i], ctr);
   i++;
 }

 printf("%d", ctr);

  return 0;
}
```

Unit 2: Final Project

You just computed final scores for all of the students in your classroom and would like to tell each student their ranking when compared to the other students in your class. For example, if Remi's score is 14, Yan's score is 2 and Maria's score is 20 then Remi's ranking is 2 since he has the second highest score, Yan's ranking is 3 since he has the lowest score and Maria's ranking is 1 since she has the highest score amongst the three students.

You'll be given the following inputs: first the number of students, then all of the students' final scores, then finally, all of the students' names. Be sure to look at the example below.

Your output should be one line per student, with each line displaying the student's name followed by "rank is" followed by the student's rank.

The order in which names are displayed in the output should be the same as the order given in the input. The ranking starts at 1 (meaning this student has the highest score), 2 for the second highest grade etc...

Note that no name has more than 50 characters and there are no more than 30 students.

Example:

Input

```
3
14
2
20
Remi
Yan
Maria
```

Output

```
Remi rank is 2
Yan rank is 3
Maria rank is 1
```

```c
#include <stdio.h>

int main(void){
    int num;
    scanf("%d", &num);

    char name[num-1][51];
    int scr[num-1];
    int cop[num-1];

    int i, j, k=0;
    int swap;

    //Saving score
    for(i=0;i<num;i++){
        scanf("%d", &scr[i]);
        cop[i] = scr[i];
    }

    //Saving name
    for(i=0;i<num;i++){
        scanf("%s", name[i]);
    }



    for(j=0;j<num;j++){
        for(i=0;i<num;i++){
            if(cop[i]<cop[i+1]){
                swap = cop[i+1];
                cop[i+1] = cop[i];
                cop[i] = swap;
            }
        }
    }

    for(i=0;i<num;i++){
        j=0;
        printf("%s rank is ", name[i]);
        while(scr[i] != cop[j]){
            j++;
        }
        printf("%d\n", j+1);
    }


    return 0;
}
```