

/* Task

You have to print the character `ch`, in the first line. Then print `s` in next line. In the last line print `sen` in the sentence.

Input Format:

First, take a character,
as input.

Then take the string, as input.

Lastly, take the sentence as input.

Output Format:

Print three lines of output. The first line prints the character,`ch`.

The second line prints the string,`s` .

The third line prints the sentence,`sen` .

*/

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{ // from here

char ch;

char s[100];

char sen[100];

scanf("%c", &ch);

scanf("%s", s);scanf("\n");

scanf("%[^\\n]*sen", sen);

printf("%c\\n", ch);

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

printf("%s\\n",sen);

//till here

return 0;

}

`/* Task`

Given a positive integer n denoting, do the following:

If n is between 1-9 both inclusive, print the lowercase English word corresponding to the number (e.g., one for 1, two for 2, etc.).

If greater than 9, print Greater than 9.

Input Format

The first line contains a single integer,

`*/``//erase everything and write below code``// Write Your Code Here``#include<stdio.h>``int main() {``int n;``scanf("%d", &n);``printf((n==1)?"one": (n==2)?"two": (n==3)?"three": (n==4)?"four": (n==5)?"five": (n==6)?"six": (n==7)?"seven": (n==8)?"eight": (n==9)?"nine": "Greater than 9");``return 0;``}`

/* In this challenge, you will use logical bitwise operators. All data is stored in its binary representation. The logical operators, and C language, use 1 to represent true and 0 to represent false. The logical operators compare bits in two numbers and return true or false, or

, for each bit compared.

Bitwise AND operator & The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0. It is denoted by &.

Bitwise OR operator | The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. It is denoted by |.

Bitwise XOR (exclusive OR) operator ^ The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by xor symbol

*/

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
// from here
int max (int n, int k){
    int and=0, or=0, xor=0;
    for (int i=1; i<n; i++){
        for (int j=i+1; j<=n; j++){
            int a = i&j, b = i|j, c = i^j;
            if (a < k && and < a){
                and = a;
            }
            if (b < k && or < b){
                or = b;
            }
            if (c < k && xor < c){
                xor = c;
            }
        }
    }
    return printf("%d\n%d\n%d", and, or, xor);
}

// till here
int main() {

    int n, k;
    scanf("%d %d", &n, &k);
    max(n,k);

    return 0;
}
```

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main()
{
    //add from here
    int a,b;
    scanf("%d %d",&a,&b);
    int sum = a+b;
    int diff = a-b;
    float a1,b1;
    scanf("%f %f",&a1,&b1);
    float sum1= a1+b1;
    //sum1=round(sum1*100)/100;
    float diff1 = a1-b1;
    //diff1=round(diff1*100)/100;
    printf("%d %d\n",sum,diff);
    printf("%.1f %.1f \n",sum1,diff1);

    //till here
    return 0;
}
```

```
int numberOfMatches(int n) {
    int matches = 0;
    #
    while (n > 1) {
        if (n % 2 == 0) {
            matches += n / 2;
            n /= 2;
        } else {
            matches += (n - 1) / 2;
            n = (n - 1) / 2 + 1;
        }
    }

    return matches;
}
```

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int main()
{ char *s[]={"null","one","two","three","four","five","six","seven","eight","nine"};
  int a, b;
  scanf("%d\\n%d", &a, &b);
  while(a<=b)
  {
    if((a<=9)&&(a>=1))
    {printf("%s\\n",s[a]);}
    else
    {
      if(a%2==0)
      {printf("even\\n");}
      else {
        printf("odd\\n");
      }
    }
    a++;
  }
  return 0;
}
```

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MIN_ELEMENT 1
#define MAX_ELEMENT 1000000
//from here
int sum (int count,...) {
    va_list ptr;
    int sum=0;
    va_start(ptr, count);
    for(int i=0;i<count;i++)
    {
        sum+=va_arg(ptr, int);
    }
    va_end(ptr);

    return sum;
}

int min(int count,...) {
    va_list ptr;
    va_start(ptr, count);
    int max=va_arg(ptr, int);

    for(int i=0;i<count;i++)
    {
        int temp = va_arg(ptr, int);
        max = temp<max ? temp : max;
    }
    return max;
}

int max(int count,...) {
    va_list ptr;
    va_start(ptr, count);
    int min=va_arg(ptr, int);

    for(int i=0;i<count;i++)
    {
        int temp = va_arg(ptr, int);
        min = temp>min ? temp : min;
    }
    return min;
}
//till here
int test_implementations_by_sending_three_elements() {
    srand(time(NULL));

    int elements[3];

    elements[0] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
    elements[1] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
    elements[2] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;

    fprintf(stderr, "Sending following three elements:\n");
    for (int i = 0; i < 3; i++) {
        fprintf(stderr, "%d\n", elements[i]);
    }

    int elements_sum = sum(3, elements[0], elements[1], elements[2]);
    int minimum_element = min(3, elements[0], elements[1], elements[2]);
    int maximum_element = max(3, elements[0], elements[1], elements[2]);

    fprintf(stderr, "Your output is:\n");
    fprintf(stderr, "Elements sum is %d\n", elements_sum);
}
```

```
fprintf(stderr, "Minimum element is %d\n", minimum_element);
fprintf(stderr, "Maximum element is %d\n\n", maximum_element);

int expected_elements_sum = 0;
for (int i = 0; i < 3; i++) {
    if (elements[i] < minimum_element) {
        return 0;
    }

    if (elements[i] > maximum_element) {
        return 0;
    }

    expected_elements_sum += elements[i];
}

return elements_sum == expected_elements_sum;
}

int test_implementations_by_sending_five_elements() {
    srand(time(NULL));

    int elements[5];

    elements[0] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
    elements[1] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
    elements[2] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
    elements[3] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
    elements[4] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;

    fprintf(stderr, "Sending following five elements:\n");
    for (int i = 0; i < 5; i++) {
        fprintf(stderr, "%d\n", elements[i]);
    }

    int elements_sum = sum(5, elements[0], elements[1], elements[2], elements[3], elements[4]);
    int minimum_element = min(5, elements[0], elements[1], elements[2], elements[3], elements[4]);
    int maximum_element = max(5, elements[0], elements[1], elements[2], elements[3], elements[4]);

    fprintf(stderr, "Your output is:\n");
    fprintf(stderr, "Elements sum is %d\n", elements_sum);
    fprintf(stderr, "Minimum element is %d\n", minimum_element);
    fprintf(stderr, "Maximum element is %d\n\n", maximum_element);

    int expected_elements_sum = 0;
    for (int i = 0; i < 5; i++) {
        if (elements[i] < minimum_element) {
            return 0;
        }

        if (elements[i] > maximum_element) {
            return 0;
        }

        expected_elements_sum += elements[i];
    }

    return elements_sum == expected_elements_sum;
}

int test_implementations_by_sending_ten_elements() {
    srand(time(NULL));

    int elements[10];
```



```

elements[0] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
elements[1] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
elements[2] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
elements[3] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
elements[4] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
elements[5] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
elements[6] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
elements[7] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
elements[8] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;
elements[9] = rand() % (MAX_ELEMENT - MIN_ELEMENT + 1) + MIN_ELEMENT;

fprintf(stderr, "Sending following ten elements:\n");
for (int i = 0; i < 10; i++) {
    fprintf(stderr, "%d\n", elements[i]);
}

int elements_sum = sum(10, elements[0], elements[1], elements[2], elements[3], elements
[4],
                        elements[5], elements[6], elements[7], elements[8], elements[9])
;
int minimum_element = min(10, elements[0], elements[1], elements[2], elements[3], eleme
nts[4],
                        elements[5], elements[6], elements[7], elements[8], elements[9])
;
int maximum_element = max(10, elements[0], elements[1], elements[2], elements[3], eleme
nts[4],
                        elements[5], elements[6], elements[7], elements[8], elements[9])
;

fprintf(stderr, "Your output is:\n");
fprintf(stderr, "Elements sum is %d\n", elements_sum);
fprintf(stderr, "Minimum element is %d\n", minimum_element);
fprintf(stderr, "Maximum element is %d\n\n", maximum_element);

int expected_elements_sum = 0;
for (int i = 0; i < 10; i++) {
    if (elements[i] < minimum_element) {
        return 0;
    }

    if (elements[i] > maximum_element) {
        return 0;
    }

    expected_elements_sum += elements[i];
}

return elements_sum == expected_elements_sum;
}

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

    while (number_of_test_cases--) {
        if (test_implementations_by_sending_three_elements()) {
            printf("Correct Answer\n");
        } else {
            printf("Wrong Answer\n");
        }

        if (test_implementations_by_sending_five_elements()) {
            printf("Correct Answer\n");
        } else {
            printf("Wrong Answer\n");
        }
    }
}

```

```
    if (test_implementations_by_sending_ten_elements()) {  
        printf("Correct Answer\n");  
    } else {  
        printf("Wrong Answer\n");  
    }  
}  
  
return 0;  
}
```

```

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* addTwoNumbers(struct ListNode* l1, struct ListNode* l2) {
    //from here
    int carry = 0;
    struct ListNode head;
    struct ListNode* cur = &head;

    while (l1 || l2 || carry) {
        int sum = carry;
        if (l1) {
            sum += l1->val;
            l1 = l1->next;
        }

        if (l2) {
            sum += l2->val;
            l2 = l2->next;
        }

        carry = sum / 10;

        cur->next = malloc(sizeof(struct ListNode));
        cur = cur->next;
        cur->val = sum % 10;
        cur->next = NULL;
    }

    return head.next;
} //till here

```

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
//Complete the following function.

int find_nth_term(int n, int a, int b, int c) {
    //from here
    switch (n) {
        case 1:
            return a;
        case 2:
            return b;
        case 3:
            return c;
        default:
            return find_nth_term(n - 1, a, b, c) + find_nth_term(n - 2, a, b, c) + find_nth_term(n - 3, a, b, c);
    }
} //till here

int main() {
    int n, a, b, c;

    scanf("%d %d %d %d", &n, &a, &b, &c);
    int ans = find_nth_term(n, a, b, c);

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

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
```

```
//Complete the following function.
```

```
int marks_summation(int* marks, int number_of_students, char gender) {
    //Write your code here.
    //from here
    int sum = 0;
    int i = (gender == 'g');

    while (i < number_of_students) {
        sum += marks[i];
        i += 2;
    }
    return sum; //till here
}

int main() {
    int number_of_students;
    char gender;
    int sum;

    scanf("%d", &number_of_students);
    int *marks = (int *) malloc(number_of_students * sizeof (int));

    for (int student = 0; student < number_of_students; student++) {
        scanf("%d", (marks + student));
    }

    scanf(" %c", &gender);
    sum = marks_summation(marks, number_of_students, gender);
    printf("%d", sum);
    free(marks);

    return 0;
}
```