```
int a;
scanf("%d", &a);
printf("%d", a);
```

```
$
```

```
int a;
scanf("%d", &a);
printf("%d", a);
```

```
$ ./a.out ↵
```

```
int a;
scanf("%d", &a);
printf("%d", a);
```

```
$ ./a.out ↵
50t ↵
```

```
int a;
scanf("%d", &a);
printf("%d", a);
```

```
$ ./a.out ↵
50t ↵
50
$
```

```c
int a, b;
scanf("%d", &a);
scanf("%d", &b);
printf("%d %d", a, b);
```

```
$ ./a.out ↵
50t ↵
50
$
```

```
int a, b;
scanf("%d", &a);
scanf("%d", &b);
printf("%d %d", a, b);
```

```
$ ./a.out ↵
50t ↵
50
$./a.out ↵
```

```
int a, b;
scanf("%d", &a);
scanf("%d", &b);
printf("%d %d", a, b);
```

```
$ ./a.out ↵
50t ↵
50
$./a.out ↵
50t ↵
```

```
int a, b;
scanf("%d", &a);
scanf("%d", &b);
printf("%d %d", a, b);
```

```
$ ./a.out ↵
50t ↵
50
$./a.out ↵
50t ↵
50 1354349328
$
```

```
int a, b, c;
scanf("%d", &a);
scanf("%d", &b);
scanf("%d", &c);
printf("%d %d %d", a, b, c);
```

```
$ ./a.out ↵
50t ↵
50
$./a.out ↵
50t ↵
50 1354349328
$
```

```
int a, b, c;
scanf("%d", &a);
scanf("%d", &b);
scanf("%d", &c);
printf("%d %d %d", a, b, c);
```

```
$ ./a.out ↵
50t ↵
50
$./a.out ↵
50t ↵
50 1354349328
$./a.out ↵
```

```
int a, b, c;
scanf("%d", &a);
scanf("%d", &b);
scanf("%d", &c);
printf("%d %d %d", a, b, c);
```

```
$ ./a.out ↵
50t ↵
50
$./a.out ↵
50t ↵
50 1354349328
$./a.out ↵
50t ↵
```

```
int a, b, c;
scanf("%d", &a);
scanf("%d", &b);
scanf("%d", &c);
printf("%d %d %d", a, b, c);
```

```
$ ./a.out ↵
50t ↵
50
$./a.out ↵
50t ↵
50 1354349328
$./a.out ↵
50t ↵
50 1354349328 1442364176
$
```

```
int a[100], i;
```

```
$
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
```

```
$
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
```

```
$
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
for (i--; i >= 0; i--)
    printf("%d\n", a[i]);
```

```
$
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
for (i--; i >= 0; i--)
    printf("%d\n", a[i]);
```

```
$ ./a.out ↵
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
for (i--; i >= 0; i--)
    printf("%d\n", a[i]);
```

```
$ ./a.out ↵
100 ↵
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
for (i--; i >= 0; i--)
    printf("%d\n", a[i]);
```

```
$ ./a.out ↵
100 ↵
-100 ↵
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
for (i--; i >= 0; i--)
    printf("%d\n", a[i]);
```

```
$ ./a.out ↵
100 ↵
-100 ↵
50 ↵
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
for (i--; i >= 0; i--)
    printf("%d\n", a[i]);
```

```
$ ./a.out ↵
100 ↵
-100 ↵
50 ↵
13 ↵
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
for (i--; i >= 0; i--)
    printf("%d\n", a[i]);
```

```
$ ./a.out ↵
100 ↵
-100 ↵
50 ↵
13 ↵
a ↵
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
for (i--; i >= 0; i--)
    printf("%d\n", a[i]);
```

```
$ ./a.out ↵
100 ↵
-100 ↵
50 ↵
13 ↵
a ↵
The numbers are :
```

```
int a[100], i;
for(i = 0; scanf("%d", &a[i]); i++)
    ;
printf("The numbers are :  \n");
for (i--; i >= 0; i--)
    printf("%d\n", a[i]);
```

```
$ ./a.out ↵
100 ↵
-100 ↵
50 ↵
13 ↵
a ↵
The numbers are :
13
50
-100
100
```

Read Binary string

Read Binary string

```
char str[20];
scanf("%[01]s", str);
printf("%s\n", str);
```

Read Binary string

```
char str[20];
scanf("%[01]s", str);
printf("%s\n", str);
```

Rules

`%[01]s` – Read only if you see 0's and 1's.

Read Binary string

```
char str[20];
scanf("%[01]s", str);
printf("%s\n", str);
```

Rules

`%[01]s` – Read only if you see `0`'s and `1`'s.

`%[abc]s` – Read only if you see `a, b, c`.

Read Binary string

```
char str[20];
scanf("%[01]s", str);
printf("%s\n", str);
```

Rules

`%[01]s` – Read only if you see `0`'s and `1`'s.

`%[abc]s` – Read only if you see `a, b, c`.

`%[0-9]s` – Read only if you digits.

Read Binary string

```
char str[20];
scanf("%[01]s", str);
printf("%s\n", str);
```

Rules

`%[01]s` – Read only if you see `0`'s and `1`'s.

`%[abc]s` – Read only if you see `a, b, c`.

`%[0-9]s` – Read only if you digits.

[...]	read till these characters are found
[∧...]	read till these characters are not found

Task

Take date as input in the following format and do not store the month.

```
01 Apr 2016
```

Task

Take date as input in the following format and do not store the month.

```
01 Apr 2016
```

Solution

```
scanf("%d  %d", &a, &b);
print("%d %d\n", a, b);
```

Task

Take date as input in the following format and do not store the month.

```
01 Apr 2016
```

Solution

```
scanf("%d %*s %d", &a, &b);
print("%d %d\n", a, b);
```

Task

Take date as input in the following format and do not store the month.

```
01 Apr 2016
```

Solution

```
scanf("%d %*s %d", &a, &b);
print("%d %d\n", a, b);
```

```
$ ./a.out ↵
```

Task

Take date as input in the following format and do not store the month.

```
01 Apr 2016
```

Solution

```
scanf("%d %*s %d", &a, &b);
print("%d %d\n", a, b);
```

```
$ ./a.out ↵
01 Mar 2016 ↵
```

Task

Take date as input in the following format and do not store the month.

```
01 Apr 2016
```

Solution

```
scanf("%d %*s %d", &a, &b);
print("%d %d\n", a, b);
```

```
$ ./a.out ↵
01 Mar 2016 ↵
1 2016
$
```

Task

Take date as input in the following format and do not store the month.

```
01 Apr 2016
```

Solution

```
scanf("%d %*s %d", &a, &b);
print("%d %d\n", a, b);
```

```
$ ./a.out ↵
01 Mar 2016 ↵
1 2016
$
```

Rule

Read the desired input, but do not store.

Task

Take date as input in the following format and do not store the month.

```
01 Apr 2016
```

Solution

```
scanf("%d %*d %d", &a, &b);
print("%d %d\n", a, b);
```

```
$ ./a.out ↵
01 Mar 2016 ↵
1 2016
$
```

Rule

Read the desired input, but do not store.

Task

Take date as input in the following format and do not store the month.

```
01 Apr 2016
```

Solution

```
scanf("%d %*f %d", &a, &b);
print("%d %d\n", a, b);
```

```
$ ./a.out ↵
01 Mar 2016 ↵
1 2016
$
```

Rule

Read the desired input, but do not store.

Another way to read character

```
scanf("%*[ \t\n]%c", &c);
printf("Character is :  %c\n", c);
```

Another way to read character

```
scanf("%*[ \t\n]%c", &c);
printf("Character is :  %c\n", c);
```

Another way to read character

```
scanf("%*[ \t\n]%c", &c);
printf("Character is :  %c\n", c);
```

Another way to read character

```
scanf("%*[ \t\n]%c", &c);
printf("Character is :  %c\n", c);
```

```
$ ./a.out ↵
```

Another way to read character

```
scanf("%*[ \t\n]%c", &c);
printf("Character is :  %c\n", c);
```

```
$ ./a.out ↵
↵
```

Another way to read character

```
scanf("%*[ \t\n]%c", &c);
printf("Character is :  %c\n", c);
```

```
$ ./a.out ↵
↵
↵
```

Another way to read character

```
scanf("%*[ \t\n]%c", &c);
printf("Character is :  %c\n", c);
```

```
$ ./a.out ↵
↵
↵
y ↵
```

Another way to read character

```
scanf("%*[ \t\n]%c", &c);
printf("Character is :  %c\n", c);
```

```
$ ./a.out ↵
↵
↵
y ↵
Character is :  y
$
```