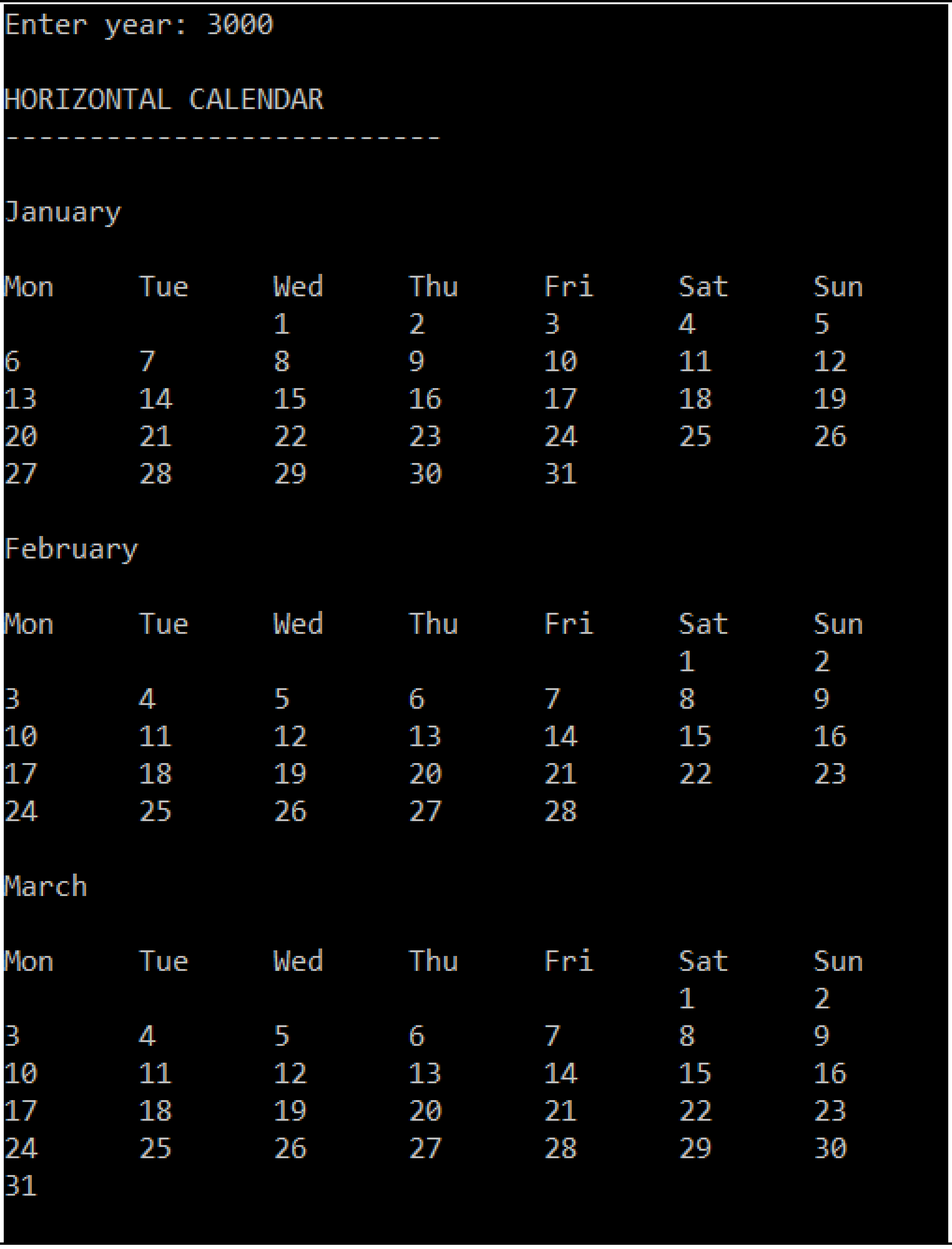
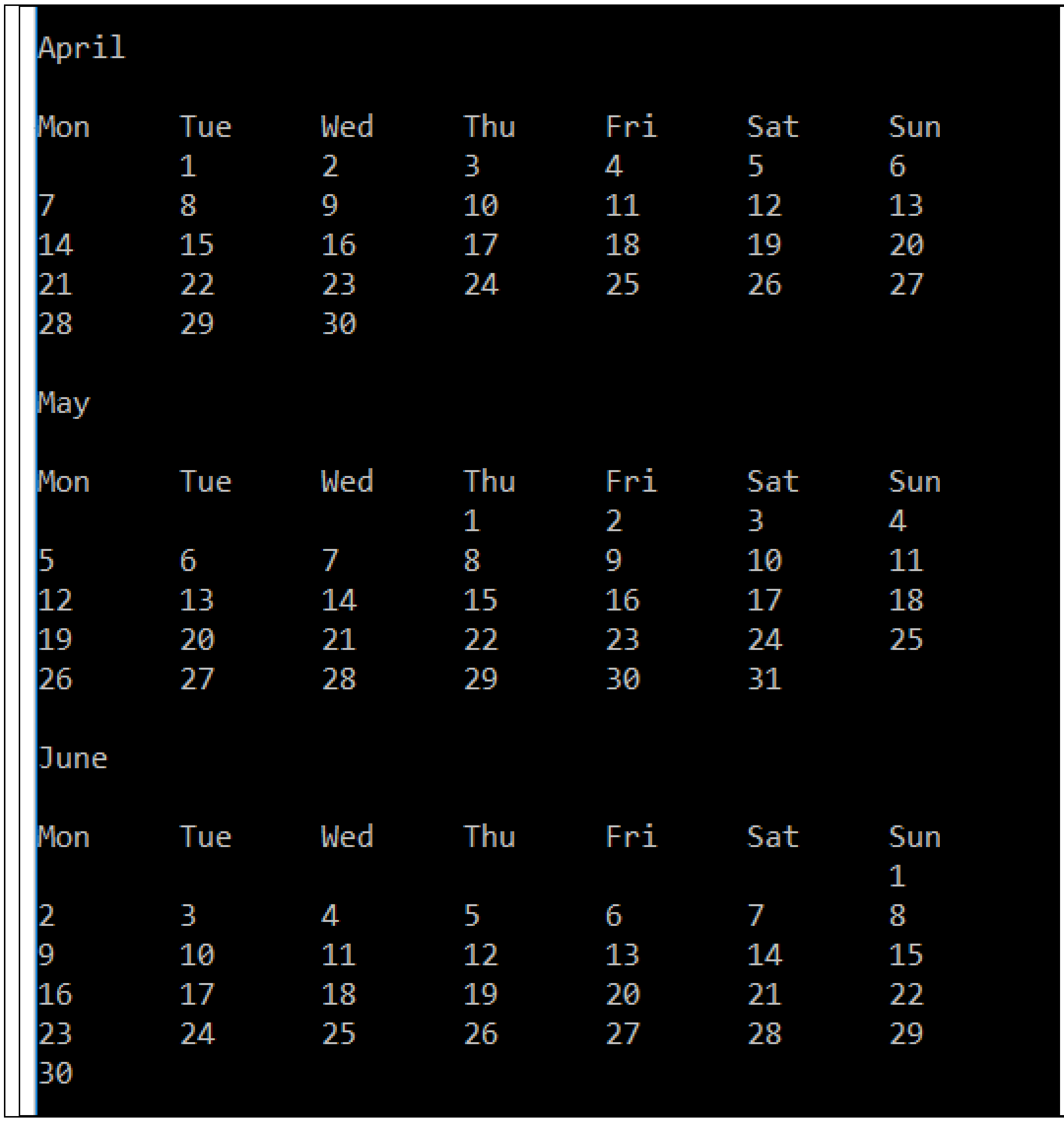
1. Make a Calendar (Horizontally as well as Vertically)

```
#include<iostream>
#include<string>
using namespace std;

bool is_leap(int year){
   return (year % 400 == 0) || (year % 100 != 0 && year % 4 == 0);
}

int print_month_horizontally(int n_days, int start_day){
   cout<<"Mon\tTue\tWed\tThu\tFri\tSat\tSun\n";
   for(int j=0; j<start_day; j++){
         cout<<"\t";
   }
   for (int i=1; i<=n_days; i++){
      cout<<i<<"\t";
      if ((i - 7 + start_day) % 7 == 0){
         cout<<"\n";
      }
   }
   int last_day = (start_day + n_days) % 7;
   return last_day;
}

int print_month_vertically(int n_days, int start_day){
   string week[7] = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
   for (int i=0; i<7; i++){
      cout<<week[i]<<"\t";
         if (i < start_day) {
            cout<<"\t";
            int a = 7-start_day +1 + i;
            while (a <= n_days){
               cout<<a<<"\t";
               a = 7+a;
            }
         }
         else
         {
            int b = i - start_day + 1;
            while (b <= n_days){
               cout<<b<<"\t";
               b = 7+b;
            }
         }
         cout<<"\n";

   }
   int last_day = (start_day + n_days) % 7;
```

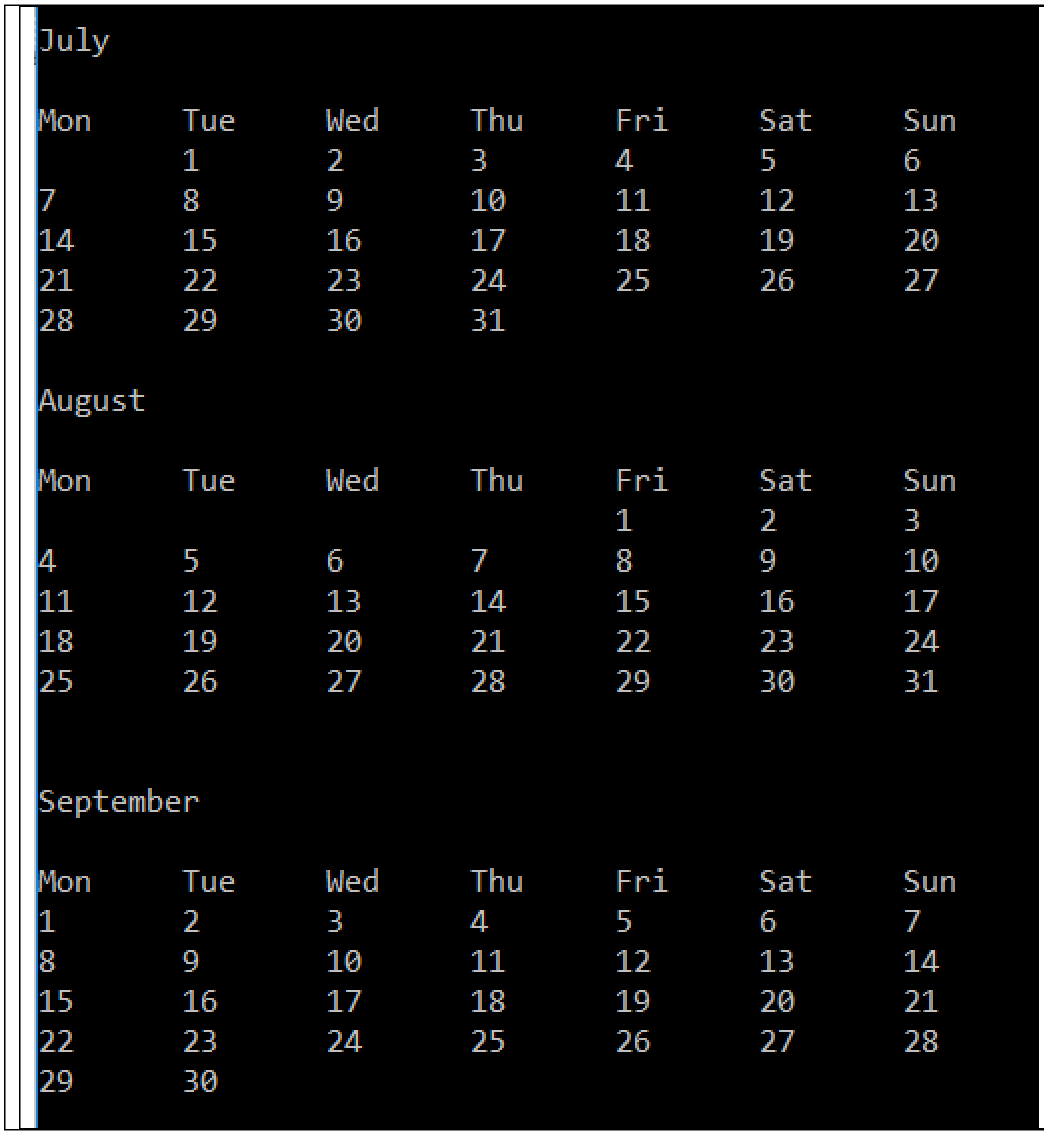
```
    return last_day;
}

int first_day(int data, int year){
    if (year >= 2020){    // since data is of 2020 (reference year) we use 2020
        int x = year - 2020;
        int counter=0;
        for (int i=2020; i<year; i++){
            if (is_leap(i)){
                counter++;
            }
        }
        int norm = x - counter;
        return (data + norm + 2*counter) % 7;
    }else if(year == 2020){
        return data;
    } else {
        int x = 2020 - year;
        int counter = 0;
        for (int i=year; i<2020; i++){
            if (is_leap(i)){
                counter++;
            }
        }
        int norm = x - counter;
        return (7 - ((data - norm - 2*counter) % 7)) % 7;
    }
}


int main(){
    int data = 2;  // 2 means wednesday which was Jan 1, 2020
    int year, feb_day=28;
    cout<<"Enter year: ";
    cin>>year;
    if (is_leap(year)){
        feb_day=29;
    }
    int first = first_day(data, year);
    cout<<"\nHORIZONTAL CALENDAR\n--------------------------";
    cout<<"\n\nJanuary\n\n";
    first = print_month_horizontally(31, first);   //this function gives the day after last date in that
month
    cout<<"\n\nFebruary\n\n";
    first = print_month_horizontally(feb_day, first);
    cout<<"\n\nMarch\n\n";
    first = print_month_horizontally(31, first);
    cout<<"\n\nApril\n\n";
    first = print_month_horizontally(30, first);
    cout<<"\n\nMay\n\n";
    first = print_month_horizontally(31, first);
    cout<<"\n\nJune\n\n";
    first = print_month_horizontally(30, first);
    cout<<"\n\nJuly\n\n";
    first = print_month_horizontally(31, first);
    cout<<"\n\nAugust\n\n";
    first = print_month_horizontally(31, first);
    cout<<"\n\nSeptember\n\n";
    first = print_month_horizontally(30, first);
    cout<<"\n\nOctober\n\n";
    first = print_month_horizontally(31, first);
    cout<<"\n\nNovember\n\n";
    first = print_month_horizontally(30, first);
    cout<<"\n\nDecember\n\n";
    first = print_month_horizontally(31, first);
    cout<<"\nVERTICAL CALENDAR\n-------------------------";
    first = first_day(data, year);
    cout<<"\n\nJanuary\n\n";
    first = print_month_vertically(31, first);   //this function gives the day after last date in that
month
    cout<<"\n\nFebruary\n\n";
    first = print_month_vertically(feb_day, first);
    cout<<"\n\nMarch\n\n";
    first = print_month_vertically(31, first);
    cout<<"\n\nApril\n\n";
    first = print_month_vertically(30, first);
    cout<<"\n\nMay\n\n";
    first = print_month_vertically(31, first);
    cout<<"\n\nJune\n\n";
    first = print_month_vertically(30, first);
    cout<<"\n\nJuly\n\n";
    first = print_month_vertically(31, first);
    cout<<"\n\nAugust\n\n";
    first = print_month_vertically(31, first);
    cout<<"\n\nSeptember\n\n";
    first = print_month_vertically(30, first);
    cout<<"\n\nOctober\n\n";
    first = print_month_vertically(31, first);
    cout<<"\n\nNovember\n\n";
    first = print_month_vertically(30, first);
    cout<<"\n\nDecember\n\n";
    first = print_month_vertically(31, first);

    return 0;
}
```



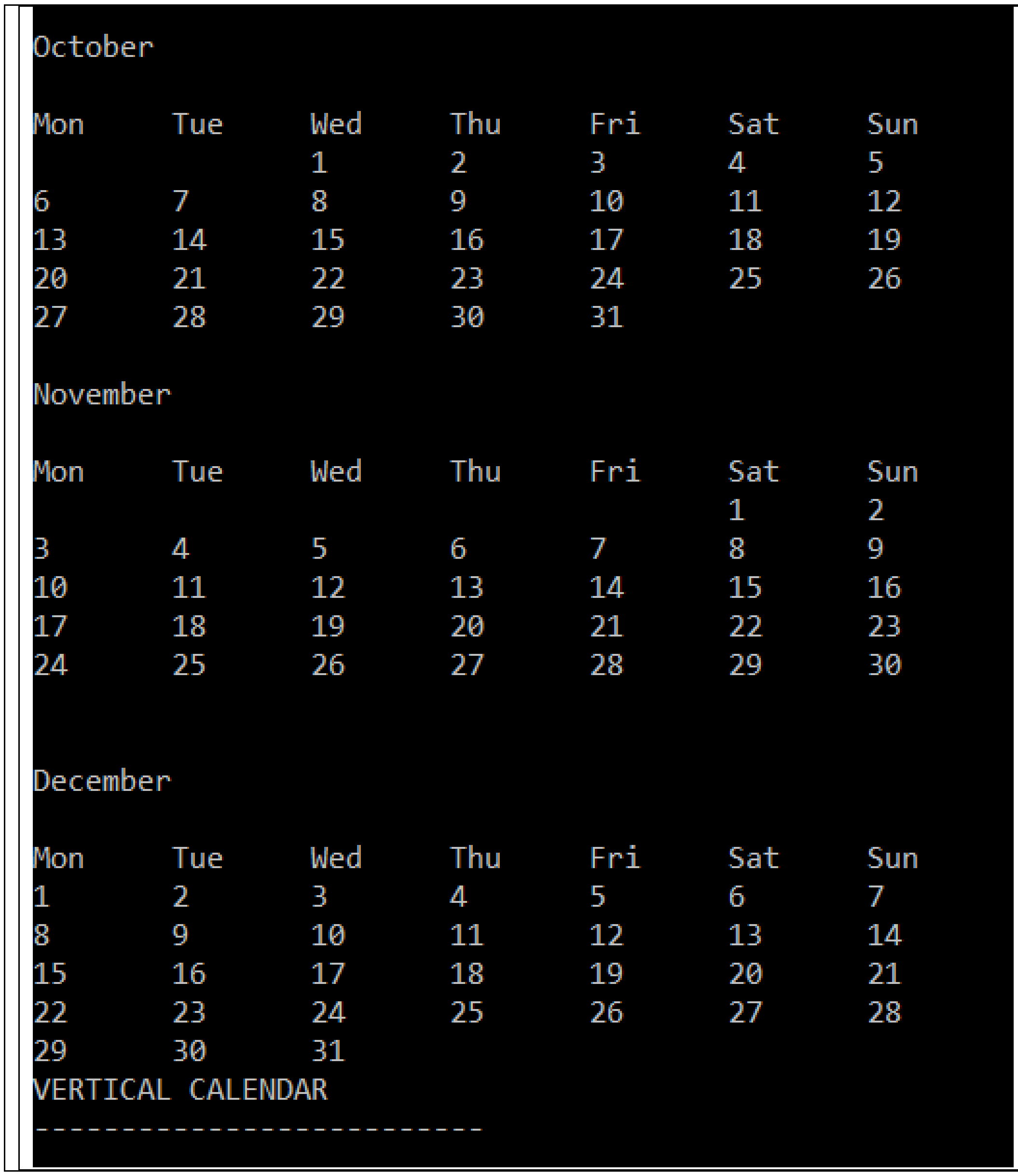
```
Enter year: 3000

HORIZONTAL CALENDAR
--------------------------

January

Mon     Tue     Wed     Thu     Fri     Sat     Sun
                1       2       3       4       5
6       7       8       9       10      11      12
13      14      15      16      17      18      19
20      21      22      23      24      25      26
27      28      29      30      31

February

Mon     Tue     Wed     Thu     Fri     Sat     Sun
                                        1       2
3       4       5       6       7       8       9
10      11      12      13      14      15      16
17      18      19      20      21      22      23
24      25      26      27      28

March

Mon     Tue     Wed     Thu     Fri     Sat     Sun
                                        1       2
3       4       5       6       7       8       9
10      11      12      13      14      15      16
17      18      19      20      21      22      23
24      25      26      27      28      29      30
31
```



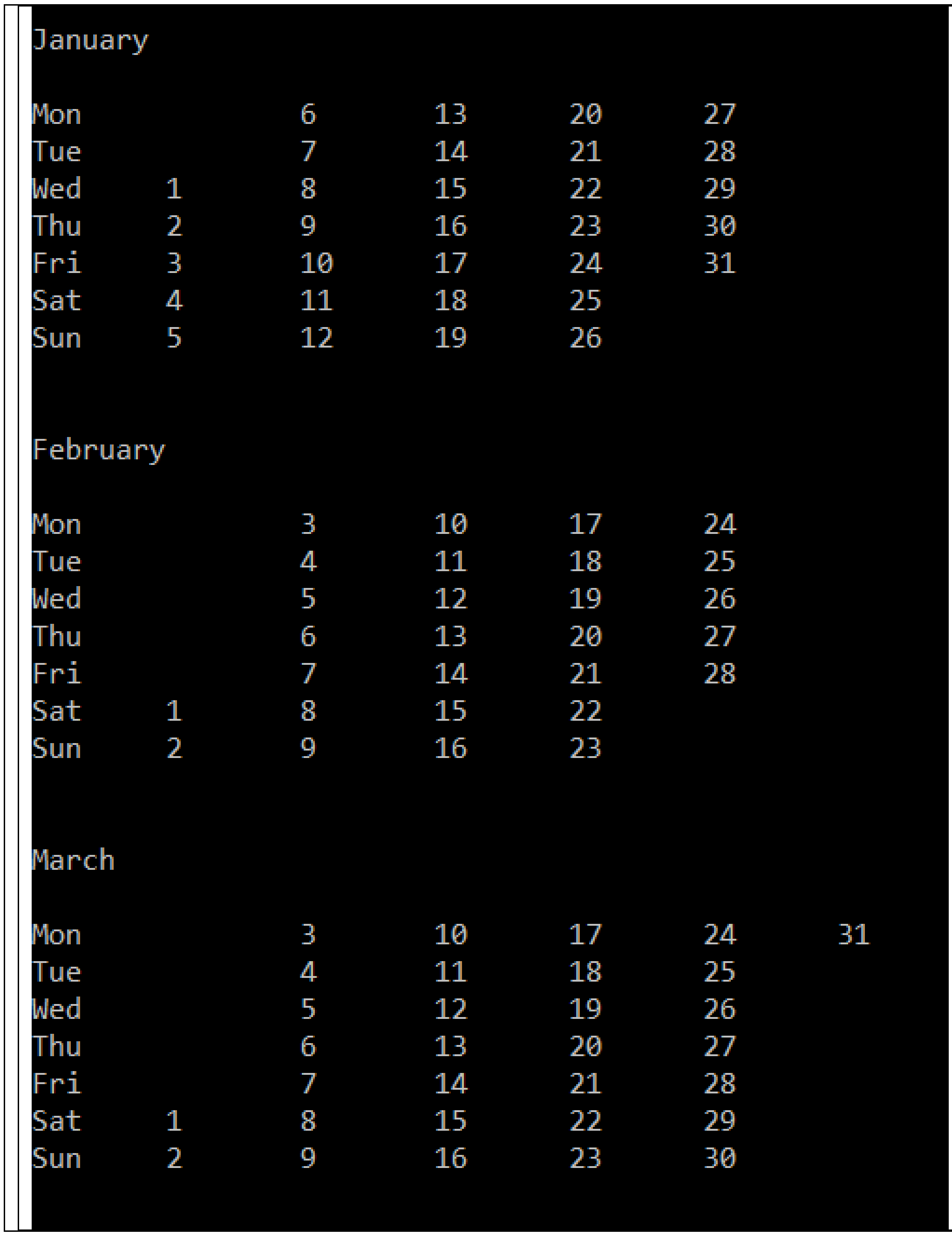
```
April

Mon     Tue     Wed     Thu     Fri     Sat     Sun
        1       2       3       4       5       6
7       8       9       10      11      12      13
14      15      16      17      18      19      20
21      22      23      24      25      26      27
28      29      30

May

Mon     Tue     Wed     Thu     Fri     Sat     Sun
                        1       2       3       4
5       6       7       8       9       10      11
12      13      14      15      16      17      18
19      20      21      22      23      24      25
26      27      28      29      30      31

June

Mon     Tue     Wed     Thu     Fri     Sat     Sun
                                                1
2       3       4       5       6       7       8
9       10      11      12      13      14      15
16      17      18      19      20      21      22
23      24      25      26      27      28      29
30
```



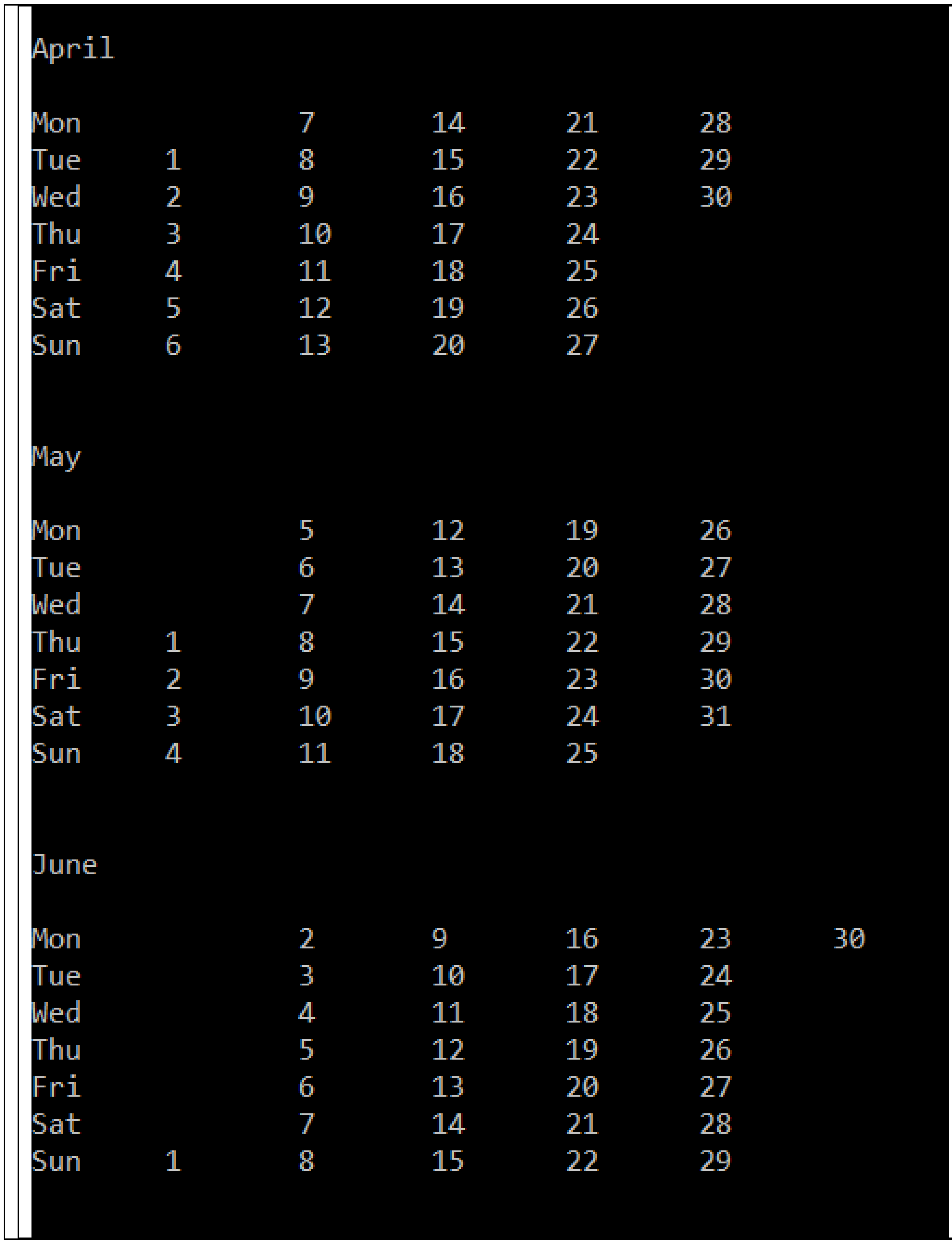
```
July

Mon     Tue     Wed     Thu     Fri     Sat     Sun
        1       2       3       4       5       6
7       8       9       10      11      12      13
14      15      16      17      18      19      20
21      22      23      24      25      26      27
28      29      30      31

August

Mon     Tue     Wed     Thu     Fri     Sat     Sun
                                1       2       3
4       5       6       7       8       9       10
11      12      13      14      15      16      17
18      19      20      21      22      23      24
25      26      27      28      29      30      31


September

Mon     Tue     Wed     Thu     Fri     Sat     Sun
1       2       3       4       5       6       7
8       9       10      11      12      13      14
15      16      17      18      19      20      21
22      23      24      25      26      27      28
29      30
```



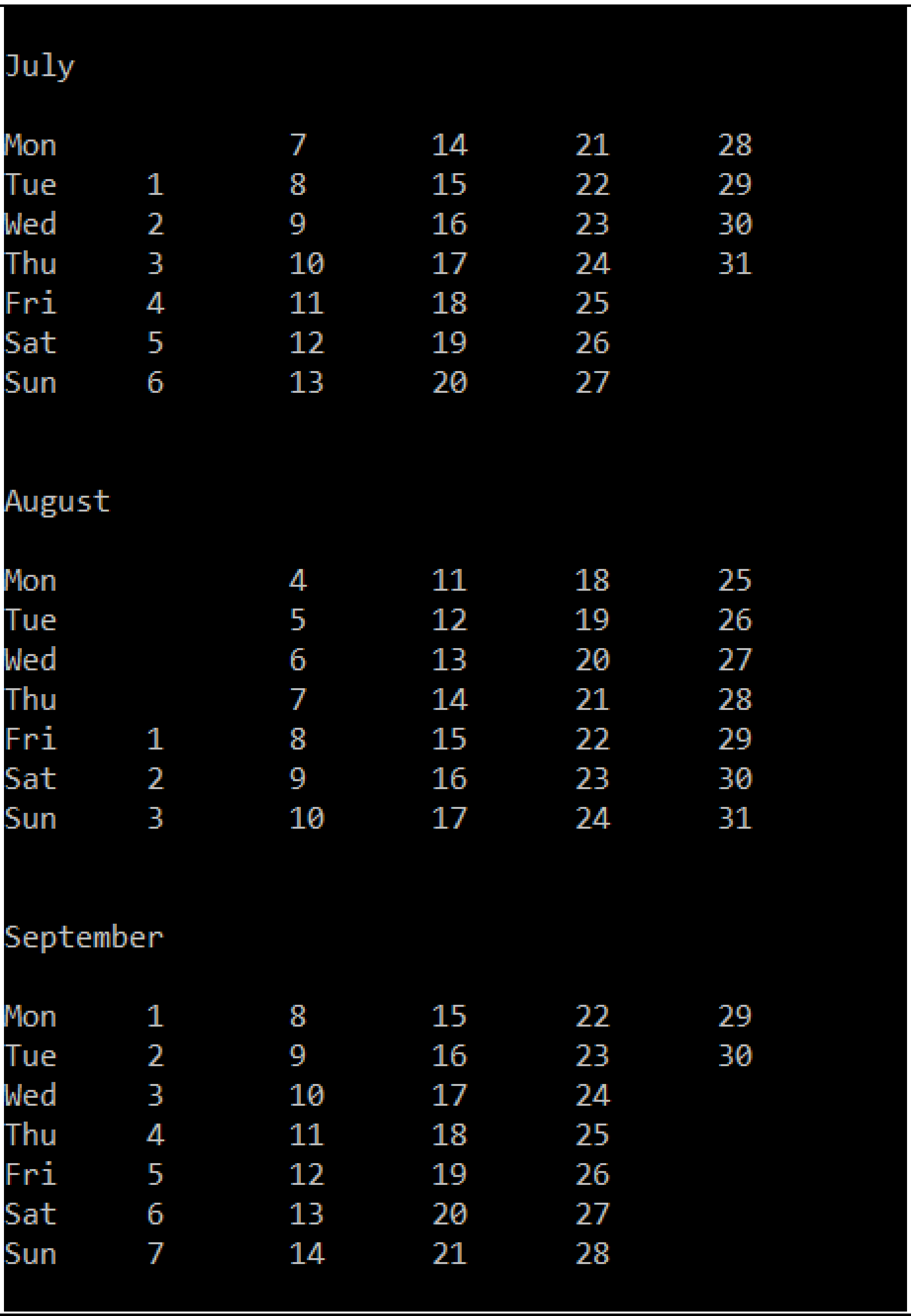
```
October

Mon     Tue     Wed     Thu     Fri     Sat     Sun
                1       2       3       4       5
6       7       8       9       10      11      12
13      14      15      16      17      18      19
20      21      22      23      24      25      26
27      28      29      30      31

November

Mon     Tue     Wed     Thu     Fri     Sat     Sun
                                        1       2
3       4       5       6       7       8       9
10      11      12      13      14      15      16
17      18      19      20      21      22      23
24      25      26      27      28      29      30


December

Mon     Tue     Wed     Thu     Fri     Sat     Sun
1       2       3       4       5       6       7
8       9       10      11      12      13      14
15      16      17      18      19      20      21
22      23      24      25      26      27      28
29      30      31
VERTICAL CALENDAR
--------------------------
```



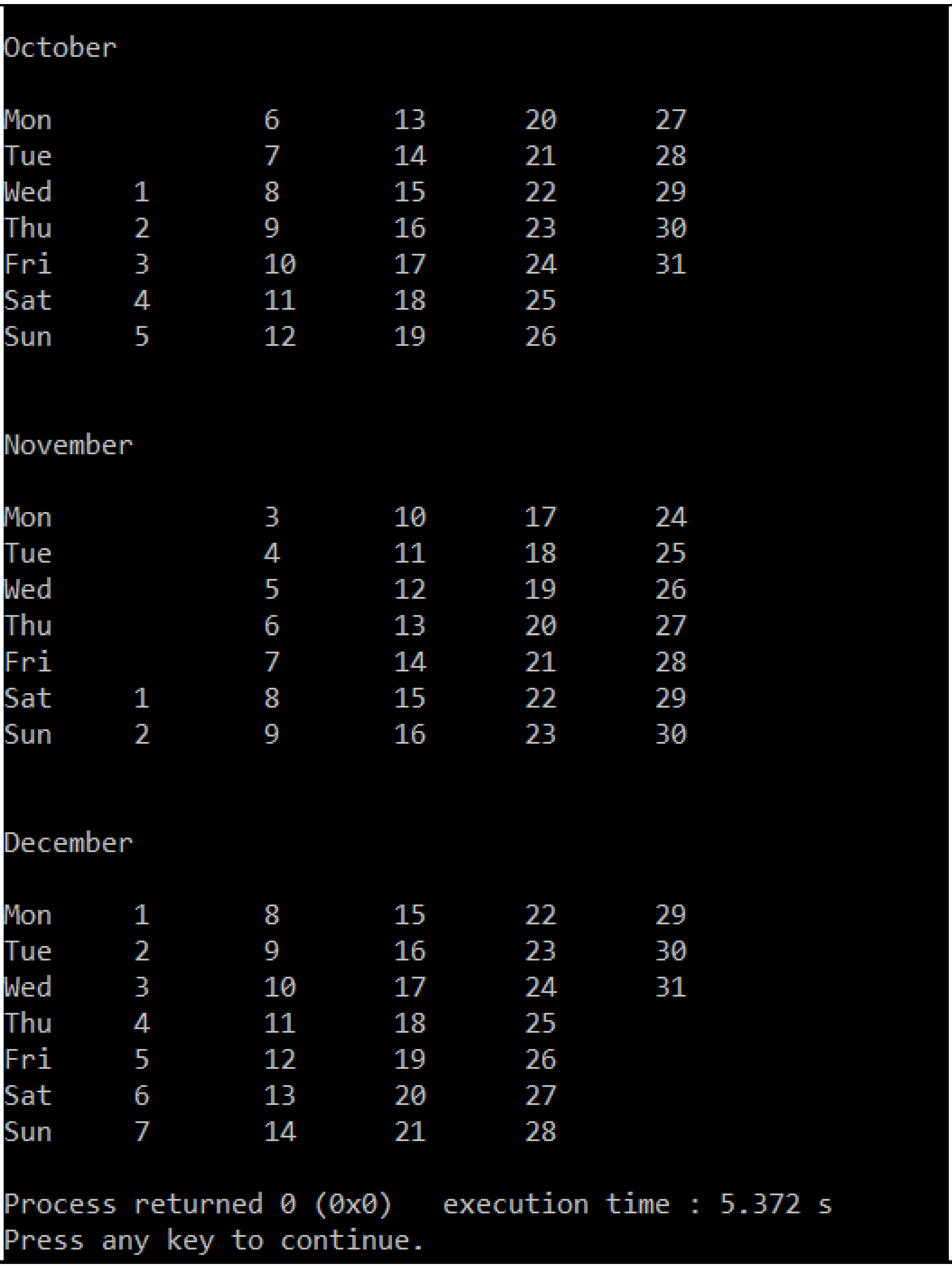
```
January

Mon             6       13      20      27
Tue             7       14      21      28
Wed     1       8       15      22      29
Thu     2       9       16      23      30
Fri     3       10      17      24      31
Sat     4       11      18      25
Sun     5       12      19      26


February

Mon             3       10      17      24
Tue             4       11      18      25
Wed             5       12      19      26
Thu             6       13      20      27
Fri             7       14      21      28
Sat     1       8       15      22
Sun     2       9       16      23


March

Mon             3       10      17      24      31
Tue             4       11      18      25
Wed             5       12      19      26
Thu             6       13      20      27
Fri             7       14      21      28
Sat     1       8       15      22      29
Sun     2       9       16      23      30
```



```
April

Mon             7       14      21      28
Tue     1       8       15      22      29
Wed     2       9       16      23      30
Thu     3       10      17      24
Fri     4       11      18      25
Sat     5       12      19      26
Sun     6       13      20      27


May

Mon             5       12      19      26
Tue             6       13      20      27
Wed             7       14      21      28
Thu     1       8       15      22      29
Fri     2       9       16      23      30
Sat     3       10      17      24      31
Sun     4       11      18      25


June

Mon             2       9       16      23      30
Tue             3       10      17      24
Wed             4       11      18      25
Thu             5       12      19      26
Fri             6       13      20      27
Sat             7       14      21      28
Sun     1       8       15      22      29
```



```
July

Mon             7       14      21      28
Tue     1       8       15      22      29
Wed     2       9       16      23      30
Thu     3       10      17      24      31
Fri     4       11      18      25
Sat     5       12      19      26
Sun     6       13      20      27


August

Mon             4       11      18      25
Tue             5       12      19      26
Wed             6       13      20      27
Thu             7       14      21      28
Fri     1       8       15      22      29
Sat     2       9       16      23      30
Sun     3       10      17      24      31


September

Mon     1       8       15      22      29
Tue     2       9       16      23      30
Wed     3       10      17      24
Thu     4       11      18      25
Fri     5       12      19      26
Sat     6       13      20      27
Sun     7       14      21      28
```

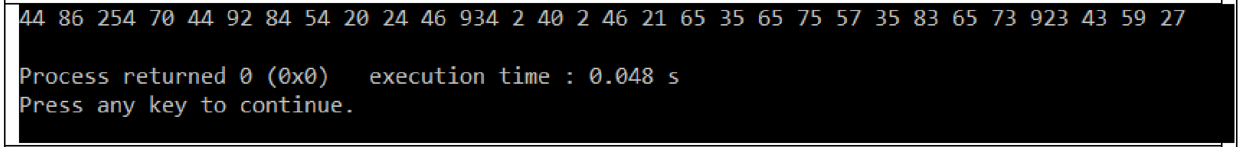


```
October

Mon             6       13      20      27
Tue             7       14      21      28
Wed     1       8       15      22      29
Thu     2       9       16      23      30
Fri     3       10      17      24      31
Sat     4       11      18      25
Sun     5       12      19      26


November

Mon             3       10      17      24
Tue             4       11      18      25
Wed             5       12      19      26
Thu             6       13      20      27
Fri             7       14      21      28
Sat     1       8       15      22      29
Sun     2       9       16      23      30


December

Mon     1       8       15      22      29
Tue     2       9       16      23      30
Wed     3       10      17      24      31
Thu     4       11      18      25
Fri     5       12      19      26
Sat     6       13      20      27
Sun     7       14      21      28

Process returned 0 (0x0)   execution time : 5.372 s
Press any key to continue.
```

2. Given array of integers, arrange the array such that even integers are on one side and odd ones in the other. Don't use another array.

Solution 1:

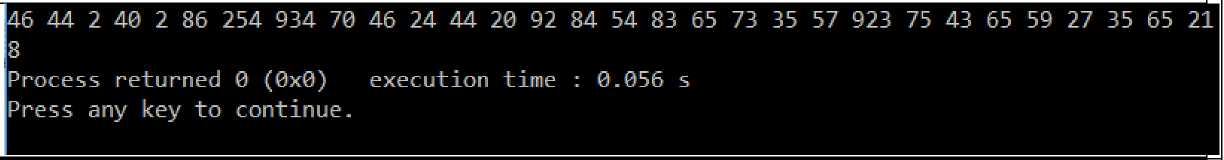
```
#include<iostream>
using namespace std;

void swape(int *x, int *y){
   int temp;
   temp = *x;
   *x = *y;
   *y = temp;
}

int main(){
   int a[30] = {21, 44, 65, 35, 65, 86, 254, 75, 70, 57, 35, 44, 83, 92, 84, 54, 20, 65, 73, 24, 46, 923,
934, 43, 2, 59 ,27, 40, 02, 46};
   int n = 30;
   int swaps = 1;
   while(swaps != 0){
      swaps = 0;
      for (int l=0, h=1; l<n-1, h<n; l++, h++){
         if (a[l] % 2 != 0 && a[h] % 2 == 0){
            swape(&a[l], &a[h]);
            swaps++;
         }
      }
   }
   for (int i=0; i<n; i++){cout<<a[i]<<" ";}
   cout<<"\n";
   return 0;
}
```



```
44 86 254 70 44 92 84 54 20 24 46 934 2 40 2 46 21 65 35 65 75 57 35 83 65 73 923 43 59 27

Process returned 0 (0x0)   execution time : 0.048 s
Press any key to continue.
```

Solution 2:

```
#include<iostream>
using namespace std;

void swape(int *x, int *y){
   int temp;
   temp = *x;
   *x = *y;
   *y = temp;
}

int main(){
   int a[30] = {21, 44, 65, 35, 65, 86, 254, 75, 70, 57, 35, 44, 83, 92, 84, 54, 20, 65, 73, 24, 46, 923,
934, 43, 2, 59 ,27, 40, 02, 46};
   int n = 30;
   // Given below is another solution which use way less swaps
   bool over = false;
   int swapes = 0;
   for (int i=0; i<n; i++){
      if (a[i] % 2 != 0){
         for (int j=n-1; j >= i; j--){
            if (a[j] % 2 == 0){
                swape(&a[i], &a[j]);
                swapes++;
                break;
            } else if (j==i) {
                over = true;
            }
        }
    }
    if (over){
        break;
    }
  }
  for (int i=0; i<n; i++){cout<<a[i]<<" ";}
  cout<<"\n"<<swapes;
  return 0;
}
```

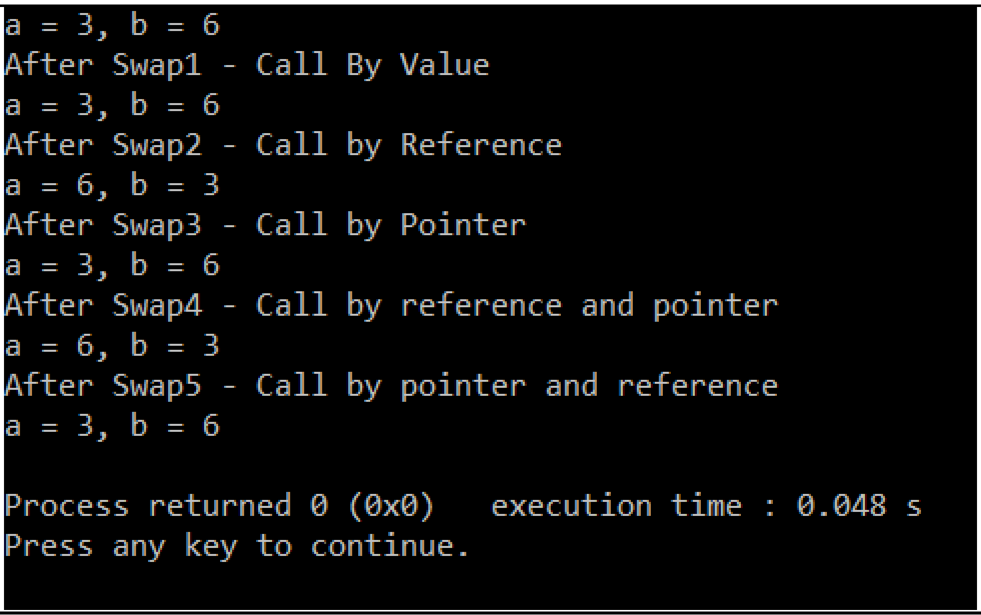


```
46 44 2 40 2 86 254 934 70 46 24 44 20 92 84 54 83 65 73 35 57 923 75 43 65 59 27 35 65 21
8
Process returned 0 (0x0)   execution time : 0.056 s
Press any key to continue.
```

3. Write the swap function in multiple ways.

```
#include<iostream>
using namespace std;

void swap1(int x, int y){ //call-by-value
  int temp = x;
  x = y;
  y = temp;
}

void swap2(int &x, int &y){ // call-by-reference     Using Alias x and y
  int temp = x;
  x = y;
  y = temp;
}

void swap3(int *x, int *y) {  // call-by-pointer
  int temp = *x;
  *x = *y;
  *y = temp;
}

void swap4(int &x, int *y) {
  int temp = x;
  x = *y;
  *y = temp;
}
```

```
void swap5(int *x, int &y) {
    int temp = *x;
    *x = y;
    y = temp;
}

// void swap6(int *x, int *y) {
//     int *temp;
//     *temp = *x;    // Assigning value to a pointer that wasn't intialised
//                    // is not encouraged however newer compilers will work
//                    // in my machine it crashed
//     *x = *y;
//     *y = *temp;
// }

int main(){
    int a=3, b=6;
    cout<<"a = "<<a<<", b = "<<b<<endl;
    swap1(a, b);     // Since it is call by value It won't swap the original a and b.
    cout<<"After Swap1 - Call By Value"<<endl;
    cout<<"a = "<<a<<", b = "<<b<<endl;
    swap2(a, b);
    cout<<"After Swap2 - Call by Reference"<<endl;
    cout<<"a = "<<a<<", b = "<<b<<endl;
    swap3(&a, &b);
    cout<<"After Swap3 - Call by Pointer"<<endl;
    cout<<"a = "<<a<<", b = "<<b<<endl;
    swap4(a, &b);
    cout<<"After Swap4 - Call by reference and pointer"<<endl;
    cout<<"a = "<<a<<", b = "<<b<<endl;
    swap5(&a, b);
    cout<<"After Swap5 - Call by pointer and reference"<<endl;
    cout<<"a = "<<a<<", b = "<<b<<endl;
    return 0;
}
```



```
a = 3, b = 6
After Swap1 - Call By Value
a = 3, b = 6
After Swap2 - Call by Reference
a = 6, b = 3
After Swap3 - Call by Pointer
a = 3, b = 6
After Swap4 - Call by reference and pointer
a = 6, b = 3
After Swap5 - Call by pointer and reference
a = 3, b = 6

Process returned 0 (0x0)   execution time : 0.048 s
Press any key to continue.
```