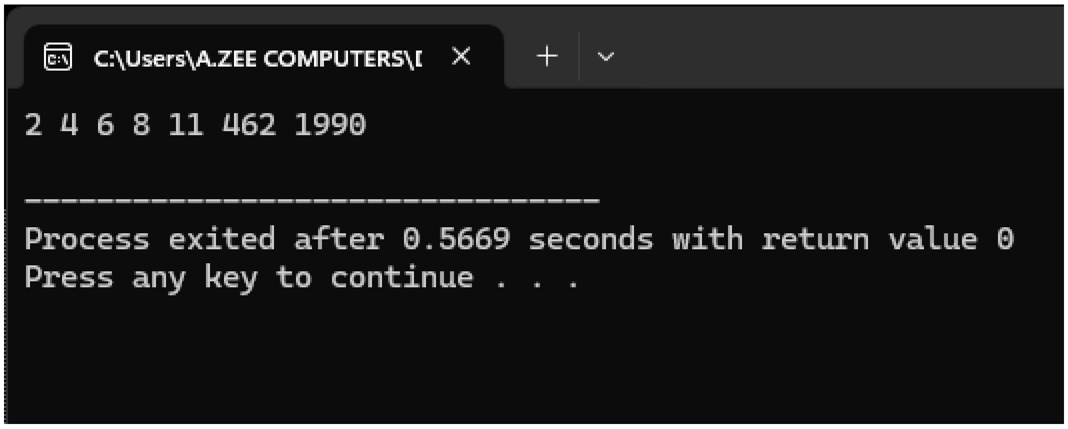
1. If the array is already sorted, we don't want to continue with the comparisons. This can be achieved with modified bubble sort. Update the code in example 02 to have a modified bubblesort function.

SOURCE CODE

```
#include <iostream>
using namespace std;
class Tamia_Lab04 {
  public:
  void ModifiedBubbleSort(int *a, int n) {
    // Ye function do cheezein leta hai: numbers ki list (a) aur kitne numbers hain (n).

     bool swapped; // Ye batata hai ke humne koi numbers swap kiye hain ya nahi.

     // Ye pehla loop numbers ki list mein jata hai.
     for (int i = 0; i < n - 1; i++) {
        swapped = false; // Hum shuru karte hain ke humne koi numbers swap nahi kiye hain ab tak.

        // Ye doosra loop har number ke pair ko check karta hai.
        for (int j = 0; j < n - i - 1; j++) {
           // Agar pehla number dusre se bara hai,
           if (a[j] > a[j + 1]) {
              swap(a[j], a[j + 1]); // Hum unhe swap karte hain taake chhota pehle aaye.
              swapped = true; // humne swap ko true kiya!
           }
        }

        // Agar humne is poore loop mein koi swap nahi kiya,
        if (!swapped) {
           break; // Hum loop ko break kardete hain qk numbers pehle se hi sorted hain!
        }
        }

        for(int i = 0; i < n; i++) {
            cout << a[i] << " ";
        }
        cout << endl;
    }
};
int main() {
    Tamia_Lab04 T;

    int array[] = {2, 4, 11, 1990, 462, 6, 8};
    int n = sizeof(array) / sizeof(array[0]);

    T.ModifiedBubbleSort(array, n);
    return 0;
}
```

OUTPUT



```
2 4 6 8 11 462 1990

--------------------------------
Process exited after 0.5669 seconds with return value 0
Press any key to continue . . .
```

2. Given an array arr[ ] of length N consisting cost of N toys and an integer K the amount with you. The task is to find maximum number of toys you can buy with K amount. Test Case: Input: N=7,K=50,arr[]={1,12,5, 111,200,1000,10},Output:4

Explanation: Thecostsofthetoys.Youcanbuyare1, 12, 5and10.

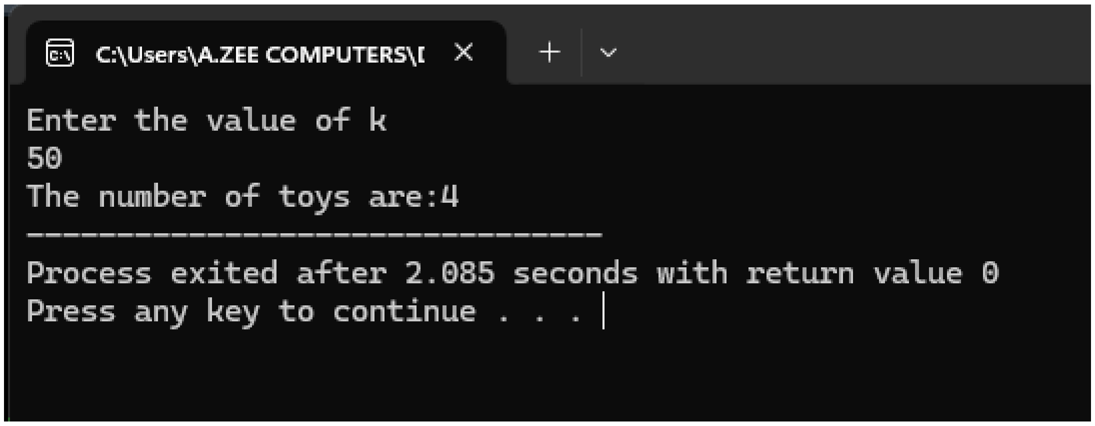
SOURCE CODE

```
#include <iostream>
using namespace std;
class Tamia_Lab04{
    public:
  void BubbleSort(int *a, int n) {
    // Ye function do cheezein leta hai: numbers ki list (a) aur kitne numbers hain (n).

    bool sorted = true; // Ye batata hai ke numbers pehle se sorted hain ya nahi.

    // Ye loop har number ko check kar raha hai.
    for (int i = 0; i < n; i++) {
      // Agar pehla number dusre se bara hai to sorted nahi hai.
      if (i == 0 && a[i] > a[i + 1]) sorted = false;
      // Agar akhri number pehle se chota hai to sorted nahi hai.
      else if (i == n - 1 && a[i] < a[i - 1]) sorted = false;
      // Agar koi number apne last se chota aur next se bara hai to sorted nahi hai.
      else if (a[i] < a[i - 1] && a[i] > a[i + 1]) sorted = false;
    }

    // Agar sab numbers pehle se sorted hain,to hum Function ko yahan se return karte hain.
    if (sorted) {
      cout << "Array is sorted";
      return;
    }
```

```
            // Agar array sorted nahi hai, to ab hum sorting shuru karte hain.
            for (int i = 0; i < n - 1; i++) { // Ye loop puri array ko check karta hai.
                for (int j = 0; j < n - i - 1; j++) { // Ye loop har number ke pair ko check karta hai.
                    // Agar pehla number dusre se bara hai,to hum unhe swap karte hain.
                    if (a[j] > a[j + 1]) {
                        swap(a[j], a[j + 1]);
                    }
                }
            }
        }

        // Ye function toys ka maximum number batata hai jo hum 'k' ke andar le sakte hain.
        int MaxNumOfToys(int *a, int k, int n) {
            int c = 0; // Hum ek counter rakhte hain jo toys ko  count karega.

            // Ye loop har toy ko check karta hai jab tak k zyada hai ya toys khatam nahi hote.
            for (int i = 0; i < n && k > 0; i++) {
                k -= a[i]; // Hum k se toy ka price km krrahe hain.
                c++;
            }

            // Agar k negative ho jata hai, to humne ek toy zyada count kiya hai.
            if (k < 0)
                c--; // Isliye hum counter ko ek se km karrahe hain.

            return c;
        }
    };
    int main(){
```

```
    Tamia_Lab04 T;
    int k;
    cout << "Enter the value of k" << endl;
    cin >> k;
    int array[] = {1,12,5, 111,200,1000,10};
    int n = sizeof(array)/sizeof(array[0]);
    T.BubbleSort(array,n);
    cout <<"The number of toys are:";
    cout << T.MaxNumOfToys(array,k,n);
    return 0;
}
```

OUTPUT



```
Enter the value of k
50
The number of toys are:4
--------------------------------
Process exited after 2.085 seconds with return value 0
Press any key to continue . . .
```

3. Create a single class Sort, which will provide the user the option to choose between all 3sorting techniques. The class should have following capabilities:

- Take an array and a string(indicating the user choice for sorting technique) as input and perform the desired sorting.

- Should allow the user to perform analysis on a randomly generated array. The analysis provides number of comparisons and number of swaps performed for each technique.

- After printing all the results in the main program, highlight the best and worst techniques.

SOURCE CODE

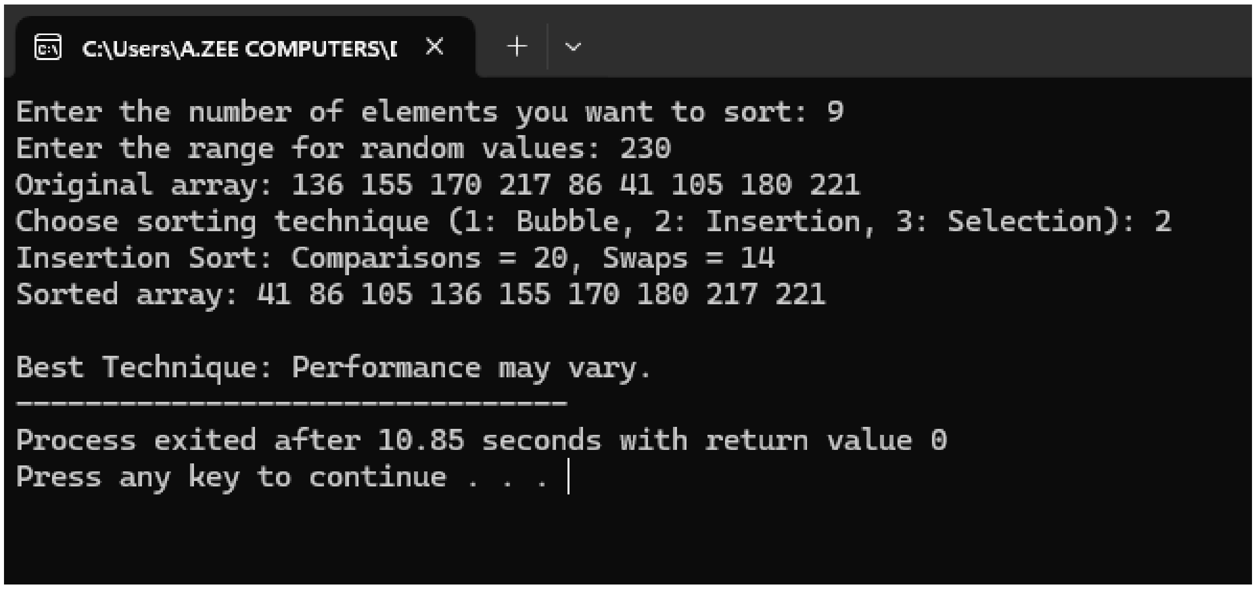
```
#include <iostream>
#include <cstdlib>
#include <ctime>
```

```
using namespace std;

class Tamia_Lab04 {
public:
    /*
    Hum 5 functions introduce karwarahe hain:
    Bubble sort, Insertion sort, Selection sort, random value, print array. Humne tamam sorting
    functions mn do new parameters comparasions or swaps ko introdue kiya h jo number of
    comparasions or swaps ko count karega take best or worst sorting technique k pata chal sake.
    */

   // Sorting techniques ke functions aur analysis
   void BubbleSort(int *a, int n, int &comparisons, int &swaps) {
      for (int i = 0; i < n - 1; i++) {
         for (int j = 0; j < n - i - 1; j++) {
            comparisons++;
            if (a[j] > a[j] + 1) {
               swap(a[j], a[j + 1]);
               swaps++;
            }
         }
      }
   }

   void InsertionSort(int *a, int n, int &comparisons, int &swaps) {
      for (int i = 1; i < n; i++) {
         int key = a[i]; // Current number ko key bana lo
         int j = i - 1;
         while (j >= 0 && a[j] > key) { // Jab tak previous number key se bara hai
            comparisons++;
            a[j + 1] = a[j]; // Shift kar do
            j--;
            swaps++;
        }
        a[j + 1] = key; // Key ko sahi jagah par daal do
        if (j >= 0) comparisons++; // Last comparison for insertion
    }
}

void SelectionSort(int *a, int n, int &comparisons, int &swaps) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i; // Minimum number ka index
        for (int j = i + 1; j < n; j++) {
            comparisons++;
            if (a[j] < a[minIndex]) { // Agar current number minimum se chota hai
                minIndex = j; // Update minimum index
            }
        }
        if (minIndex != i) {
            swap(a[minIndex], a[i]);
            swaps++;
        }
    }
}

void randomValue(int *a, int n, int range) {
    srand(static_cast<unsigned int>(time(0)));
    for (int i = 0; i < n; i++) {
        a[i] = rand() % range;
    }
```

```
        }

        void printArray(int *a, int n) {
            for (int i = 0; i < n; i++) {
                cout << a[i] << " ";
            }
            cout << endl;
        }
    };

    int main() {
        Tamia_Lab04 T;
        int n, range;

        cout << "Enter the number of elements you want to sort: ";
        cin >> n;
        cout << "Enter the range for random values: ";
        cin >> range;

        int *array = new int[n]; // Dynamically array ke liye memory allocate karo
        T.randomValue(array, n, range);

        cout << "Original array: ";
        T.printArray(array, n);
        int comparisons = 0, swaps = 0;

        cout << "Choose sorting technique (1: Bubble, 2: Insertion, 3: Selection): ";
        int choice;
        cin >> choice;
```

```
switch (choice) {
    case 1:
        T.BubbleSort(array, n, comparisons, swaps);
        cout << "Bubble Sort: Comparisons = " << comparisons << ", Swaps = " << swaps << endl;
        break;
    case 2:
        comparisons = 0;
        swaps = 0;
        T.InsertionSort(array, n, comparisons, swaps);
        cout << "Insertion Sort: Comparisons = " << comparisons << ", Swaps = " << swaps << endl;
        break;
    case 3:
        comparisons = 0;
        swaps = 0;
        T.SelectionSort(array, n, comparisons, swaps);
        cout << "Selection Sort: Comparisons = " << comparisons << ", Swaps = " << swaps << endl;
        break;
    default:
        cout << "Invalid choice!" << endl;
        delete[] array;
        return 1;
}

cout << "Sorted array: ";
T.printArray(array, n);

// Comparison analysis
cout << "\nBest Technique: ";
if (comparisons == 0) {
    cout << "No comparisons made.";
    } else if (comparisons <= 10) {
        cout << "Best performance achieved!";
    } else {
        cout << "Performance may vary.";
    }
    delete[] array;
    return 0;
}
```

OUTPUT



```
Enter the number of elements you want to sort: 9
Enter the range for random values: 230
Original array: 136 155 170 217 86 41 105 180 221
Choose sorting technique (1: Bubble, 2: Insertion, 3: Selection): 2
Insertion Sort: Comparisons = 20, Swaps = 14
Sorted array: 41 86 105 136 155 170 180 217 221

Best Technique: Performance may vary.
--------------------------------
Process exited after 10.85 seconds with return value 0
Press any key to continue . . .
```

4. Given an array of integers arr ,sort the array by performing a series of pancake flips. In one pancake flip we do the followingsteps:

- Choose an integer k where1 <=k<=arr.length.

- Reverse the sub-array arr[0...k-1](0-indexed).

For example, if arr = [3,2,1,4] and we performed a pancake flip choosing k = 3, we reverse the sub-array [3,2,1], so arr =[1,2,3,4] after the pancakeflip at k =3.Return an array of the k- values corresponding to sequence of pancake flips that sort arr .Any valid answer that sorts the array within 10*arr.length flips will be judged as correct. Example1:Input:arr=[3,2,4,1],Output:[4,2,4,3]

Explanation: We perform 4pancake flips,with k-values 4,2,4,and3. Starting state:arr=[3,2,4,1]

After 1st flip(k =4):arr=[1,4,2,3]

After 2ndflip (k= 2):arr=[4,1,2, 3]

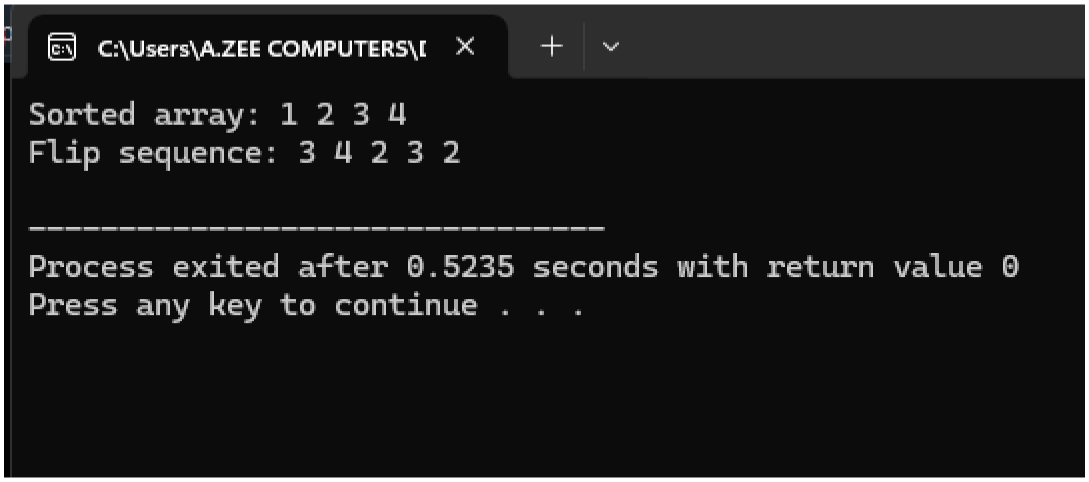
After 3rd flip(k =4): arr=[3,2, 1,4] After4thflip(k=3):arr=[1,2,3, 4],which issorted.

SOURCE CODE

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
class Tamia_004 {
public:
    vector<int> pancake(vector<int>& a) {
        vector<int> answer;
        int n = a.size();

        // Yeh loop poore array ko reverse karke sort karega
        for (int j = n; j > 1; j--) {
            // Current size tak maximum element ka index dhoondte hain
            int max_index = findMaxIndex(a, j);

            // Agar maximum element apni sahi jagah par nahi hai
            if (max_index != j - 1) {
                // Agar maximum element pehle se aage nahi hai toh pehle usay aage le aate hain
                if (max_index != 0) {
                    flip(a, max_index + 1); // Flip karte hain
                    answer.push_back(max_index + 1); // k-value ko result mein dalte hain
                }

                // Ab maximum element ko sahi position par le aate hain
                flip(a, j);
                answer.push_back(j); // Aur uska k-value bhi result mein daal dete hain
            }
        }

        return answer;
    }

private:
    // Yeh function maximum element ka index dhoondta hai
    int findMaxIndex(vector<int>& a, int n) {
        int max_index = 0; // Pehle se assume karte hain pehla element maximum hai
        for (int i = 1; i < n; i++) {
            if (a[i] > a[max_index]) {
                max_index = i;
            }
        }
        return max_index;
    }

    // Yeh function array ko 0 se k-1 tak reverse karta hai
    void flip(vector<int>& arr, int k) {
        reverse(arr.begin(), arr.begin() + k);
    }
};

int main() {
    Tamia_004 T;
    vector<int> a = {3, 2, 4, 1};
    vector<int> flips = T.pancake(a);
    cout << "Sorted array: ";
    for (int num : a) {
        cout << num << " ";
```

```
}
    cout << endl;
    cout << "Flip sequence: ";
    for (int flip : flips) {
        cout << flip << " ";
}
    cout << endl;
    return 0;
}
```

OUTPUT



```
Sorted array: 1 2 3 4
Flip sequence: 3 4 2 3 2

--------------------------------
Process exited after 0.5235 seconds with return value 0
Press any key to continue . . .
```

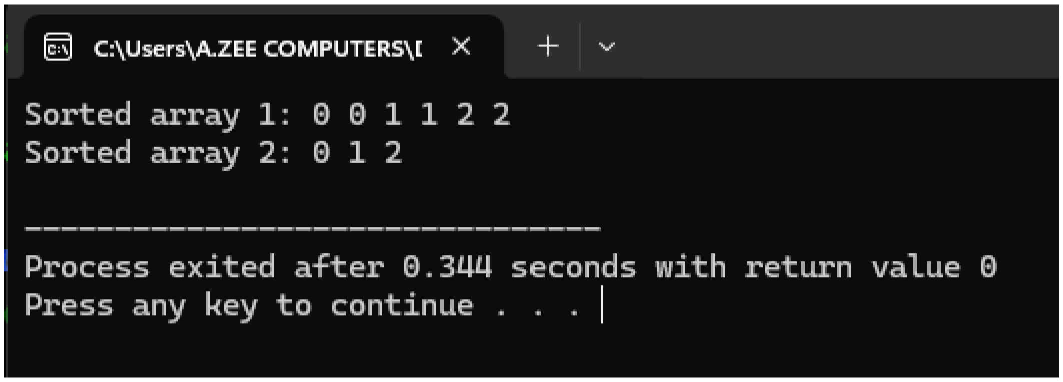
5. Given an array nums with n objects colored red, white, or blue, sort them inplace so thatobjects of the same color are adjacent, with the colors in the order red, white, and blue. We will use the integers 0,1,and2 to represent the colorred,white,andblue,respectively.You must solve this problem by writing a sort function. Example1:Input:nums=[2,0,2,1,1,0],Output:[0,0,1,1,2,2] Example2:Input:nums=[2,0,1],Output:[0,1,2]

SOURCE CODE

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

class Tamia_004 {
public:
    // we are bubble sort algorithm.
    void bubbleSort(int nums[], int n) {
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (nums[j] > nums[j + 1]) {
                    swap(nums[j], nums[j + 1]);
                }
            }
        }
    }
};

int main() {
    Tamia_004 T;
    int nums1[] = {2, 0, 2, 1, 1, 0};
    int nums2[] = {2, 0, 1};
    int size1 = sizeof(nums1) / sizeof(nums1[0]);
    int size2 = sizeof(nums2) / sizeof(nums2[0]);
    T.bubbleSort(nums1, size1);
    T.bubbleSort(nums2, size2);
    cout << "Sorted array 1: ";
    for (int i = 0; i < size1; i++) {
        cout << nums1[i] << " ";
    }
    cout << endl;
    cout << "Sorted array 2: ";
    for (int i = 0; i < size2; i++) {
        cout << nums2[i] << " ";
    }
    cout << endl;
    return 0;
```

}

OUTPUT



C:\Users\A.ZEE COMPUTERS\

```
Sorted array 1: 0 0 1 1 2 2
Sorted array 2: 0 1 2

--------------------------------
Process exited after 0.344 seconds with return value 0
Press any key to continue . . .
```