DATA STRUCTURES AND ALGORITHMS

CYCLESHEET-3

NAME: ANUBHAV JAIN
REG.NO: 22BIT0210
FACULTY: PROF VIJAYAN.E

Data Structures and Algorithms

Cyclesheet-3

Name: Anubhav Jain

Reg.No:22BIT0210

1.Write a program to implement bubble sort

Source code:

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

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

void Bubble(int A[],int n)
{
    int i,j,flag=0;
    for(i=0;i<n-1;i++)
    {
        flag=0;
        for(j=0;j<n-i-1;j++)
        {
            if(A[j]>A[j+1])
            {
                swap(&A[j],&A[j+1]);
                flag=1;
            }
        }
        if(flag==0)
            break;
    }
}
int main()
```

```
{
    int n,i;
    printf("Enter number of elements in array:");
    scanf("%d",&n);
    int A[n];
    printf("Enter %d integers :",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&A[i]);
    }
    Bubble(A,n);
    printf("Sorted Array: \n");
    for(i=0;i<n;i++)
    {
        printf("%d ",A[i]);
    }
    printf("\n");
}
```

Output:

```
C:\Users\rajiv\OneDrive\Desk

Enter number of elements in array:5
Enter 5 integers :34
66
33
55
22
Sorted Array:
22 33 34 55 66

Process returned 0 (0x0)   execution time : 44.445 s
Press any key to continue.
```

Question 2:

2. **Write a program to implement selection sort**

Source code:

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

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

void SelectionSort(int A[], int n)
{
    int i, j, k;
    for (i = 0; i < n - 1; i++)
    {
        for (j = k = i; j < n; j++)
        {
            if (A[j] < A[k])
                k = j;
        }
        swap(&A[i], &A[k]);
    }
}

int main()
{
    int n, i;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int *A = (int *)malloc(n * sizeof(int));

    if (A == NULL)
    {
        printf("Memory allocation failed.\n");
        return 1;
```

```
    }

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &A[i]);
    }

    SelectionSort(A, n);

    printf("Sorted array:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d ", A[i]);
    }
    printf("\n");
    free(A);

    return 0;
}
```

Output:

```
C:\Users\rajiv\OneDrive\Desk

Enter the number of elements: 5
Enter 5 elements:
22
8
46
88
57
Sorted array:
8 22 46 57 88

Process returned 0 (0x0)   execution time : 8.092 s
Press any key to continue.
```

Question 3:

3. Write a program to implement insertion sort

Source code:

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

void Insertion(int A[],int n)
{
    int i,j,x;
    for(i=1;i<n;i++)
    {
        j=i-1;
        x=A[i];
        while(j>-1 && A[j]>x)
        {
            A[j+1]=A[j];
            j--;
        }
        A[j+1]=x;
    }
}
int main()
{
    int n,i;
    printf("Enter number of elements in array:");
    scanf("%d",&n);
    int A[n];
    printf("Enter %d integers :",n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&A[i]);
    }
    Insertion(A,n);
    printf("Sorted Array: \n");
    for(i=0;i<n;i++)
    {
        printf("%d ",A[i]);
    }
    printf("\n");
}
```

OUTPUT:

```
C:\Users\rajiv\OneDrive\Desk

Enter number of elements in array:5
Enter 5 integers :58
22
66
88
122
Sorted Array:
22 58 66 88 122

Process returned 0 (0x0)   execution time : 9.318 s
Press any key to continue.
```

Question 4. **Write a program to implement merge sort**

Source code:

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

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

void Merge(int A[], int l, int mid, int h) {
    int i = l, j = mid + 1, k = l;
    int B[100];

    while (i <= mid && j <= h) {
        if (A[i] < A[j])
            B[k++] = A[i++];
        else
            B[k++] = A[j++];
    }

    for (; i <= mid; i++)
        B[k++] = A[i];

    for (; j <= h; j++)
        B[k++] = A[j];

    for (i = l; i <= h; i++)
        A[i] = B[i];
}

void MergeSort(int A[], int l, int h) {
    int mid;
    if (l < h) {
        mid = (l + h) / 2;
        MergeSort(A, l, mid);
        MergeSort(A, mid + 1, h);
        Merge(A, l, mid, h);
    }
}

int main() {
```

```c
    int n, i;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    int A[n];

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }

    MergeSort(A, 0, n - 1);

    printf("Sorted array:\n");
    for (i = 0; i < n; i++)
        printf("%d ", A[i]);

    printf("\n");

    return 0;
}
```

OUTPUT:

```c
#include <stdio.h>
#include <stdlib.h>

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

void Merge(int A[], int l, int mid, int h) {
    int i = l, j = mid + 1, k = l;
    int B[100];

    while (i <= mid && j <= h) {
        if (A[i] < A[j])
            B[k++] = A[i++];
        else
            B[k++] = A[j++];
    }

    for (; i <= mid; i++)
        B[k++] = A[i];

    for (; j <= h; j++)
        B[k++] = A[j];

    for (i = l; i <= h; i++)
        A[i] = B[i];
}

void MergeSort(int A[], int l, int h) {
    int mid;
```

```
C:\Users\rajiv\OneDrive\Desk

Enter the number of elements in the array: 5
Enter 5 elements:
84
22
66
44
100
Sorted array:
22 44 66 84 100

Process returned 0 (0x0)   execution time : 10.671 s
Press any key to continue.
```

Question 5: **Write a program to implement quick sort**

SOURCE CODE:

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

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

int partition(int A[], int l, int h) {
    int pivot = A[h];
    int i = (l - 1);

    for (int j = l; j <= h - 1; j++) {
        if (A[j] <= pivot) {
            i++;
            swap(&A[i], &A[j]);
        }
    }
    swap(&A[i + 1], &A[h]);
    return (i + 1);
}

void QuickSort(int A[], int l, int h) {
    if (l < h) {
        int j = partition(A, l, h);
        QuickSort(A, l, j - 1);
        QuickSort(A, j + 1, h);
    }
}

int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int A[n];

    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
```

```
        scanf("%d", &A[i]);
    }

    QuickSort(A, 0, n - 1);

    printf("Sorted array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");

    return 0;
}
```

OUTPUT:

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

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

int partition(int A[], int l, int h) {
    int pivot = A[h];
    int i = (l - 1);

    for (int j = l; j <= h - 1; j++) {
        if (A[j] <= pivot) {
            i++;
            swap(&A[i], &A[j]);
        }
    }
    swap(&A[i + 1], &A[h]);
    return (i + 1);
}

void QuickSort(int A[], int l, int h) {
    if (l < h) {
        int j = partition(A, l, h);
        QuickSort(A, l, j - 1);
        QuickSort(A, j + 1, h);
    }
}

int main() {
```

```
C:\Users\rajiv\OneDrive\Desk

Enter the number of elements: 5
Enter 5 elements:
35
78
86
93
32
Sorted array:
32 35 78 86 93

Process returned 0 (0x0)   execution time : 18.344 s
Press any key to continue.
```

Question 6: **Write a program to implement linear search**

Source Code:

```
#include <stdio.h>

struct Array
{
    int A[10];
    int size;
    int length;
};

void Display(struct Array arr)
{
    int i;
    printf("\nElements are\n");
    for (i = 0; i < arr.length; i++)
        printf("%d ", arr.A[i]);
}

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

int LinearSearch(struct Array *arr, int key)
{
    int i;
    for (i = 0; i < arr->length; i++)
    {
        if (key == arr->A[i])
        {
            swap(&arr->A[i], &arr->A[0]);
            return i;
        }
    }
    return -1;
}

int main()
{
    struct Array arr1;
```

```
    int i;

    printf("Enter the length of the array (up to 10): ");
    scanf("%d", &arr1.length);

    if (arr1.length > 10 || arr1.length < 1)
    {
        printf("Invalid length. Please enter a length between 1 and 10.\n");
        return 1;
    }

    printf("Enter %d elements for the array:\n", arr1.length);
    for (i = 0; i < arr1.length; i++)
    {
        scanf("%d", &arr1.A[i]);
    }
    arr1.size = 10;

    int key;
    printf("Enter the value to search for: ");
    scanf("%d", &key);

    int result = LinearSearch(&arr1, key);

    if (result != -1)
    {
        printf("Element found at index %d\n", result);
    }
    else
    {
        printf("Element not found in the array.\n");
    }

    Display(arr1);
    return 0;
}
```

OUTPUT:

```
C:\Users\rajiv\OneDrive\Desk

Enter the length of the array (up to 10): 5
Enter 5 elements for the array:
4
35
22
66
99
Enter the value to search for: 99
Element found at index 4

Elements are
99 35 22 66 4
Process returned 0 (0x0)   execution time : 10.060 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk

Enter the length of the array (up to 10): 5
Enter 5 elements for the array:
62
56
22
77
88
Enter the value to search for: 100
Element not found in the array.

Elements are
62 56 22 77 88
Process returned 0 (0x0)   execution time : 9.460 s
Press any key to continue.
```

Question 7: **Write a program to implement binary search**

SOURCE CODE:

```
#include <stdio.h>

struct Array
{
    int A[10];
    int size;
    int length;
};

void Display(struct Array arr)
{
    int i;
    printf("\nElements are\n");
    for (i = 0; i < arr.length; i++)
        printf("%d ", arr.A[i]);
}

int BinarySearch(struct Array arr, int key)
{
    int l, mid, h;
    l = 0;
    h = arr.length - 1;
    while (l <= h)
    {
        mid = (l + h) / 2;
        if (key == arr.A[mid])
            return mid;
        else if (key < arr.A[mid])
            h = mid - 1;
        else
            l = mid + 1;
    }
    return -1;
}

int main()
{
    struct Array arr1;
    int i, key;

    printf("Enter the length of the array (up to 10): 
");
```

```
    scanf("%d", &arr1.length);

    if (arr1.length > 10 || arr1.length < 1)
    {
        printf("Invalid length. Please enter a length
between 1 and 10.\n");
        return 1;
    }

    printf("Enter %d sorted elements for the array:\n",
arr1.length);
    for (i = 0; i < arr1.length; i++)
    {
        scanf("%d", &arr1.A[i]);
    }
    arr1.size = 10;

    printf("Enter the value to search for: ");
    scanf("%d", &key);

    int result = BinarySearch(arr1, key);

    if (result != -1)
    {
        printf("Element found at index %d\n", result);
    }
    else
    {
        printf("Element not found in the array.\n");
    }

    Display(arr1);
    return 0;
}
```

Output:

binary.c - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Start here | linear.c | binary.c | CQsort.c | traversal.c | *bst.c

```c
struct Array
{
    int A[10];
    int size;
    int length;
};

void Display(struct Array arr)
{
    int i;
    printf("\nElements are\n");
    for (i = 0; i < arr.length; i++)
        printf("%d ", arr.A[i]);
}

int BinarySearch(struct Array arr, int key)
{
    int l, mid, h;
    l = 0;
    h = arr.length - 1;
    while (l <= h)
    {
        mid = (l + h) / 2;
        if (key == arr.A[mid])
            return mid;
        else if (key < arr.A[mid])
            h = mid - 1;
        else
            l = mid + 1;
    }
    return -1;
}
```

C:\Users\rajiv\OneDrive\Desk

```
Enter the length of the array (up to 10): 5
Enter 5 sorted elements for the array:
63
78
22
34
55
Enter the value to search for: 55
Element found at index 4

Elements are
63 78 22 34 55
Process returned 0 (0x0)   execution time : 13.885 s
Press any key to continue.
```

binary.c - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Start here | linear.c | binary.c | CQsort.c | traversal.c | *bst.c

```c
struct Array
{
    int A[10];
    int size;
    int length;
};

void Display(struct Array arr)
{
    int i;
    printf("\nElements are\n");
    for (i = 0; i < arr.length; i++)
        printf("%d ", arr.A[i]);
}

int BinarySearch(struct Array arr, int key)
{
    int l, mid, h;
    l = 0;
    h = arr.length - 1;
    while (l <= h)
    {
        mid = (l + h) / 2;
        if (key == arr.A[mid])
            return mid;
        else if (key < arr.A[mid])
            h = mid - 1;
        else
            l = mid + 1;
    }
    return -1;
}
```

C:\Users\rajiv\OneDrive\Desk

```
Enter the length of the array (up to 10): 5
Enter 5 sorted elements for the array:
34
62
667
3
66
Enter the value to search for: 35
Element not found in the array.

Elements are
34 62 667 3 66
Process returned 0 (0x0)   execution time : 11.415 s
Press any key to continue.
```

Question 8: **Write a program to implement binary tree traversals**

Source Code:

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

struct TreeNode {
    int data;
    struct TreeNode* left;
    struct TreeNode* right;
};

struct TreeNode* createNode(int value) {
    struct TreeNode* newNode = (struct
TreeNode*)malloc(sizeof(struct TreeNode));
    if (!newNode) {
        perror("Memory allocation failed");
        exit(EXIT_FAILURE);
    }
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

struct TreeNode* insertNode(struct TreeNode* root, int
value) {
    if (root == NULL) {
        return createNode(value);
    }

    if (value < root->data) {
        root->left = insertNode(root->left, value);
    } else if (value > root->data) {
        root->right = insertNode(root->right, value);
    }

    return root;
}

// Function for inorder traversal
void inorderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        printf("%d ", root->data);
        inorderTraversal(root->right);
```

```
    }
}

void preorderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorderTraversal(root->left);
        preorderTraversal(root->right);
    }
}

void postorderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        postorderTraversal(root->left);
        postorderTraversal(root->right);
        printf("%d ", root->data);
    }
}

int main() {
    struct TreeNode* root = NULL;
    int n, value;

    printf("Enter the number of elements in the binary
tree: ");
    scanf("%d", &n);

    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        root = insertNode(root, value);
    }

    printf("Inorder Traversal: ");
    inorderTraversal(root);
    printf("\n");

    printf("Preorder Traversal: ");
    preorderTraversal(root);
    printf("\n");

    printf("Postorder Traversal: ");
    postorderTraversal(root);
    printf("\n");
```

```
    return 0;
}
```

OUTPUT:

```
C:\Users\rajiv\OneDrive\Desk

Enter the number of elements in the binary tree: 7
Enter the elements: 35
66
22
77
29
46
26
Inorder Traversal: 22 26 29 35 46 66 77
Preorder Traversal: 35 22 29 26 66 46 77
Postorder Traversal: 26 29 22 46 77 66 35

Process returned 0 (0x0)   execution time : 10.936 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk

Enter the number of elements in the binary tree: 5
Enter the elements: 25
77
27
26
0
Inorder Traversal: 0 25 26 27 77
Preorder Traversal: 25 0 77 27 26
Postorder Traversal: 0 26 27 77 25

Process returned 0 (0x0)   execution time : 8.232 s
Press any key to continue.
```

Question 9: **Write a program to implement binary search tree**

Source Code:

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

struct TreeNode {
    int data;
    struct TreeNode* left;
    struct TreeNode* right;
};


struct TreeNode* createNode(int value) {
    struct TreeNode* newNode = (struct
TreeNode*)malloc(sizeof(struct TreeNode));
    if (!newNode) {
        perror("Memory allocation failed");
        exit(EXIT_FAILURE);
    }
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

struct TreeNode* insertNode(struct TreeNode* root, int
value) {
    if (root == NULL) {
        return createNode(value);
    }

    if (value < root->data) {
        root->left = insertNode(root->left, value);
    } else if (value > root->data) {
        root->right = insertNode(root->right, value);
    }

    return root;
}

struct TreeNode* searchNode(struct TreeNode* root, int
value) {
    if (root == NULL || root->data == value) {
        return root;
```

```
    }

    if (value < root->data) {
        return searchNode(root->left, value);
    } else {
        return searchNode(root->right, value);
    }
}

void inorderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        printf("%d ", root->data);
        inorderTraversal(root->right);
    }
}

int main() {
    struct TreeNode* root = NULL;
    int n, value;

    printf("Enter the number of elements in the binary
search tree: ");
    scanf("%d", &n);

    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        root = insertNode(root, value);
    }

    printf("Inorder Traversal (sorted): ");
    inorderTraversal(root);
    printf("\n");

    printf("Enter a value to search: ");
    scanf("%d", &value);

    struct TreeNode* result = searchNode(root, value);
    if (result) {
        printf("Value %d found in the tree.\n", value);
    } else {
        printf("Value %d not found in the tree.\n",
value);
    }}
```

OUTPUT:

```
linear.c  binary.c  CQsort.c  traversa
#include <stdio.h>
#include <stdlib.h>

struct TreeNode {
    int data;
    struct TreeNode* left;
    struct TreeNode* right;
};


struct TreeNode* createNode(int
    struct TreeNode* newNode = (
    if (!newNode) {
        perror("Memory allocatic
        exit(EXIT_FAILURE);
    }
    newNode->data = value;
    newNode->left = newNode->rig
    return newNode;
}

struct TreeNode* insertNode(stru
    if (root == NULL) {
        return createNode(value)
    }

    if (value < root->data) {
```

```
C:\Users\rajiv\OneDrive\Desk
Enter the number of elements in the binary search tree: 5
Enter the elements: 63
22
89
45
100
Inorder Traversal (sorted): 22 45 63 89 100
Enter a value to search: 89
Value 89 found in the tree.

Process returned 0 (0x0)   execution time : 18.801 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk
Enter the number of elements in the binary search tree: 5
Enter the elements: 45
77
89
43
29
Inorder Traversal (sorted): 29 43 45 77 89
Enter a value to search: 100
Value 100 not found in the tree.

Process returned 0 (0x0)   execution time : 42.063 s
Press any key to continue.
```

Challenging Question:

Challenging Question:
Assume in the Regional Passport Office, a multitude of applicants arrive each day for passport renewal. A list is maintained in the database to store the renewed passports arranged in the increased order of passport ID. The list already would contain the records renewed till the previous day. Apply Insertion sort technique to place the current day's records in the list.
Later the office personnel wish to sort the records based on the date of renewal so as to know the count of renewals done each day. Taking into consideration the fact that each record has several fields (around 25 fields), follow Selection sort logic to implement the same.

Source Code:

```
#include<stdio.h>
struct Date
{
int year,month,day;
};
struct passp
{
 int id;
 struct Date d;
};
int compare(const struct Date d1,const struct Date d2)
{
 if (d1.year< d2.year)
 return 1;
 if (d1.year == d2.year && d1.month < d2.month)
 return 1;
 if (d1.year == d2.year && d1.month == d2.month &&
 d1.day < d2.day)

 return 1;
 return 0;
}
int main()
{
int i,j,min,n;
struct passp t;
struct passp p1[100],p2[100];
printf("Enter number of records: ");
```

```
scanf("%d",&n);
printf("Enter records: \n");
for(i=0;i<n;i++)
 {
printf("\npassport id: ");
scanf("%d",&p1[i].id);
printf("date of renewal: ");
scanf("%d%d%d",&p1[i].d.day,&p1[i].d.month,&p1[i].d.yea
r);
 }
 for(i=0;i<n;i++)
 {
 p2[i]=p1[i];
 }
for (i=1;i<n;i++){
t=p1[i];
j=i-1;

while(p1[j].id > t.id && j>=0)
{
p1[j+1]=p1[j];
j--;
}
p1[j+1]=t;
}
printf("\n-----------------------------------------------
-----\n");
printf("records sorted based on passport id::\n");
for (i=0;i<n;i++)
{
printf("\nid: ");
printf("%d",p1[i].id);
printf("\t\t\tdate: ");
 printf("%d%s%d%s%d",p1[i].d.day,"-",p1[i].d.month,"-
",p1[i].d.year);
}
printf("\n-----------------------------------------------
-----\n");
for (i=0;i<n;i++)
{
min=i;
for(j=i+1;j<n;j++)
{
if(compare(p2[j].d,p2[min].d))
{
```

```
min=j;

}
}
t=p2[i];
p2[i]=p2[min];
p2[min]=t;
}
printf("records sorted based on date of renewal::\n");

for (i=0;i<n;i++){
 printf("\ndate: ");
 printf("%d%s%d%s%d",p2[i].d.day,"-",p2[i].d.month,"-
",p2[i].d.year);
 printf("\t\t\tid: ");
 printf("%d",p2[i].id);
}
}
```

Output:

```
C:\Users\rajiv\OneDrive\Desk

Enter number of records: 5
Enter records:

passport id: 100
date of renewal: 22-10-2024

passport id: 101
date of renewal: 27-11-2027

passport id: 102
date of renewal: 23-8-2024

passport id: 103
date of renewal: 19-11-2026

passport id: 104
date of renewal: 24-12-2025

--------------------------------------------------
records sorted based on passport id::

id: 100                 date: 22--10--2024
id: 101                 date: 27--11--2027
id: 102                 date: 23--8--2024
id: 103                 date: 19--11--2026
id: 104                 date: 24--12--2025
--------------------------------------------------
records sorted based on date of renewal::

date: 27--11--2027                      id: 101
date: 19--11--2026                      id: 103
date: 24--12--2025                      id: 104
date: 22--10--2024                      id: 100
date: 23--8--2024                       id: 102
Process returned 0 (0x0)   execution time : 83.909 s
Press any key to continue.
```

Output:

```
C:\Users\rajiv\OneDrive\Desk

Enter number of records: 3
Enter records:

passport id: 3432552
date of renewal: 22-10-2019

passport id: 263663
date of renewal: 19-11-2027

passport id: 636363
date of renewal: 23-10-2026

--------------------------------------------------
records sorted based on passport id::

id: 263663                      date: 19--11--2027
id: 636363                      date: 23--10--2026
id: 3432552                     date: 22--10--2019
--------------------------------------------------
records sorted based on date of renewal::

date: 19--11--2027                      id: 263663
date: 23--10--2026                      id: 636363
date: 22--10--2019                      id: 3432552
Process returned 0 (0x0)   execution time : 26.271 s
Press any key to continue.
```