**ADVANCED ALGORITHMS**



**NAME- PRANAY KUMAR**

**ENROLLMENT-2024MSCS010**

**M.Sc (COMPUTER SCIENCE)**

**ADVANCED ALGORITHMS**

**INDEX:-**

|  |  |
| --- | --- |
| **1** | **LINEAR SEARCH** |
| **2** | **BINARY SEARCH** |
| **3** | **BUBBLE SORT WITHOUT FLAG** |
| **4** | **BUBBLE SORT WITH FLAG** |
| **5** | **MERGE SORT** |
| **6** | **QUICK SORT** |
| **7** | **NON- DETERMINISTIC LINEAR SEARCH** |
| **8** | **FRACTIONAL KNAPSACK USING GREEDY ALGORITHMS** |
| **9** | **FRACTIONAL KNAPSACK NON DETRMINISTIC** |
| **10** | **NON- DETERMINITIC PRIMALITY TEST** |
| **11** | **COUNTING SORT** |
| **12** | **SUM OF SUBSET PROBLEM** |
| **13** | **N QUEEN PROBLEMS** |
| **14** | **RANDOMMISSED QUICK SORT** |
| **15** | Comparison between different sorting |

**Q1) WRITE A PROGRAM IN LINEAR SEARCH**

**ANS.**

**#include<stdio.h>**

**int linearsearch(int arr[], int i,int n,int element){**

**for(i=0;i<n;i++){**

**if(arr[i]==element){**

**return i;**

**}**

**}**

**return -1;**

**}**

**int main(){**

**int array[50],i,n,element;**

**printf("Enter size of array:");**

**scanf("%d",&n);**

**for(i=0;i<n;i++){**

**scanf("%d",&array[i]);**

**}**

**printf("Enter value to find");**

**scanf("%d",&element);**

**int result = linearsearch(array, i,n,element);**

**if (result == -1)**

**printf("Not found");**

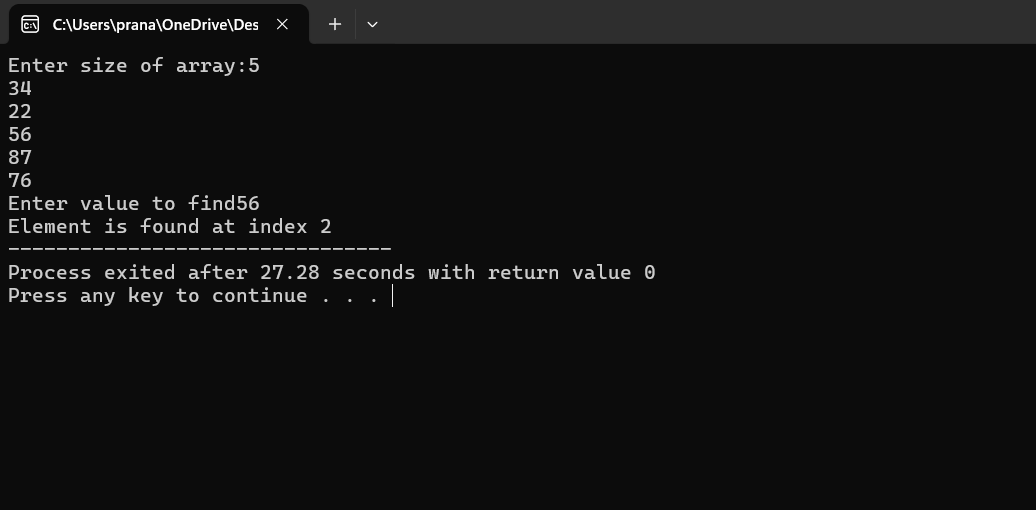
**else**

**printf("Element is found at index %d", result);**

**return 0;**

**}**

**OUTPUT.**

****

***Q2) WRITE A PROGRAM IN BINARY SEARCH:***

**ANS-**

**#include <stdio.h>**

**int binarySearch(int array[], int x, int low, int high) {**

**while (low <= high) {**

**int mid = (low + high )/ 2;**

**if (array[mid] == x)**

**return mid;**

**else if(array[mid] < x)**

**low = mid + 1;**

**else**

**high = mid - 1;**

**}**

**return -1;**

**}**

**int main() {**

**int array[50],i,n,x;**

**printf("Enter size of array:");**

**scanf("%d",&n);**

**for(i=0;i<n;i++){**

**scanf("%d",&array[i]);**

**}**

**printf("Enter value to find");**

**scanf("%d",&x);**

**int result = binarySearch(array, x, 0, n - 1);**

**if (result == -1)**

**printf("Not found");**

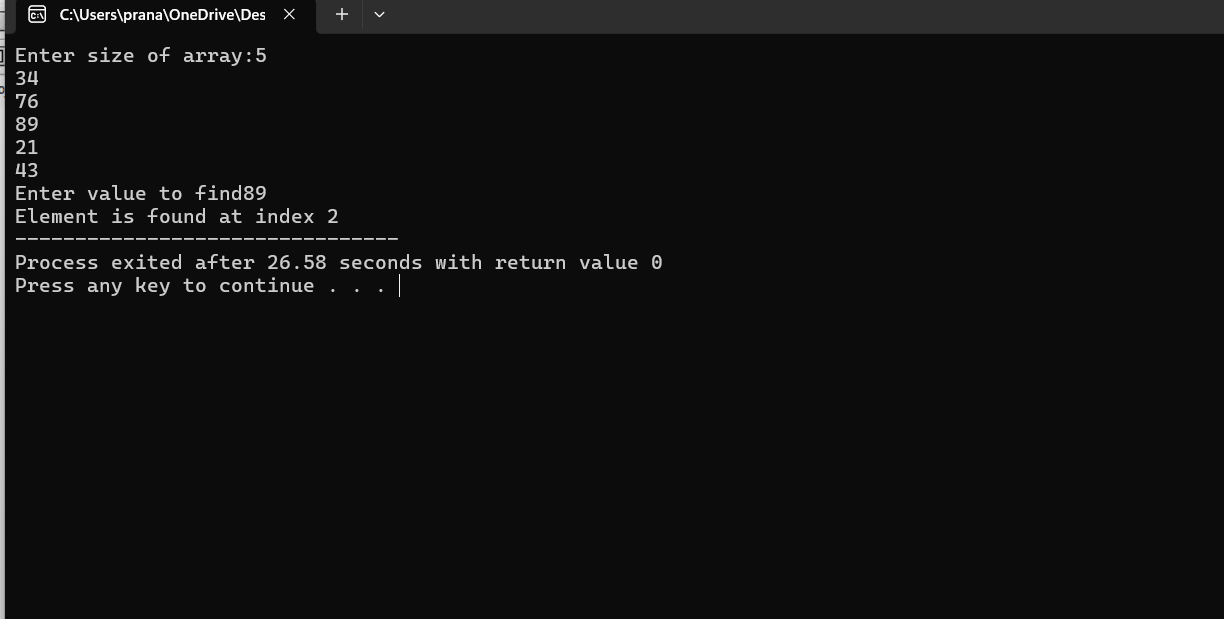
**else**

**printf("Element is found at index %d", result);**

**return 0;**

**}**

**OUTPUT.**

****

**Q3) WRITE A PROGRAM IN BUBBLE SORT WITHOUT FLAG-**

**ANS-**

**#include <stdio.h>**

**void bubbleSort(int arr[], int n) {**

**int i, j;**

**int temp;**

**for (i = 0; i < n - 1; i++) {**

**for (j = 0; j < n - i - 1; j++) {**

**if (arr[j] > arr[j + 1]) {**

**temp = arr[j];**

**arr[j] = arr[j + 1];**

**arr[j + 1] = temp;**

**}**

**}**

**}**

**}**

**void printArray(int arr[], int size) {**

**for (int i = 0; i < size; i++) {**

**printf("%d ", arr[i]);**

**}**

**printf("\n");**

**}**

**int main() {**

**int n;**

**printf("Enter the number of elements: ");**

**scanf("%d", &n);**

**int arr[n];**

**printf("Enter %d elements: \n", n);**

**for (int i = 0; i < n; i++) {**

**scanf("%d", &arr[i]);**

**}**

**printf("Unsorted array: \n");**

**printArray(arr, n);**

**bubbleSort(arr, n);**

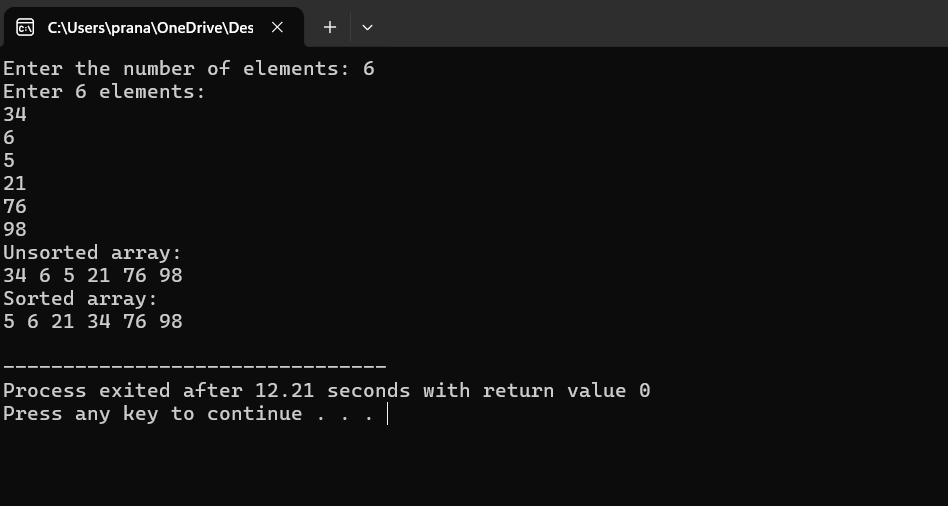
**printf("Sorted array: \n");**

**printArray(arr, n);**

**return 0;**

**}**

**OUTPUT;**

****

**Q4) WRITE A PROGRAM IN BUBBLE SORT WITH FLAG-**

**ANS-**

**#include <stdio.h>**

**void bubbleSort(int arr[], int n) {**

**int i, j, temp;**

**int swapped;**

**for (i = 0; i < n-1; i++) {**

**swapped = 0;**

**for (j = 0; j < n-i-1; j++) {**

**if (arr[j] > arr[j+1]) {**

**temp = arr[j];**

**arr[j] = arr[j+1];**

**arr[j+1] = temp;**

**swapped = 1;**

**}**

**}**

**if (swapped == 0)**

**break;**

**}**

**}**

**int main() {**

**int n, i;**

**printf("Enter number of elements: ");**

**scanf("%d", &n);**

**int arr[n];**

**printf("Enter %d elements: ", n);**

**for (i = 0; i < n; i++) {**

**scanf("%d", &arr[i]);**

**}**

**bubbleSort(arr, n);**

**printf("Sorted array: ");**

**for (i = 0; i < n; i++) {**

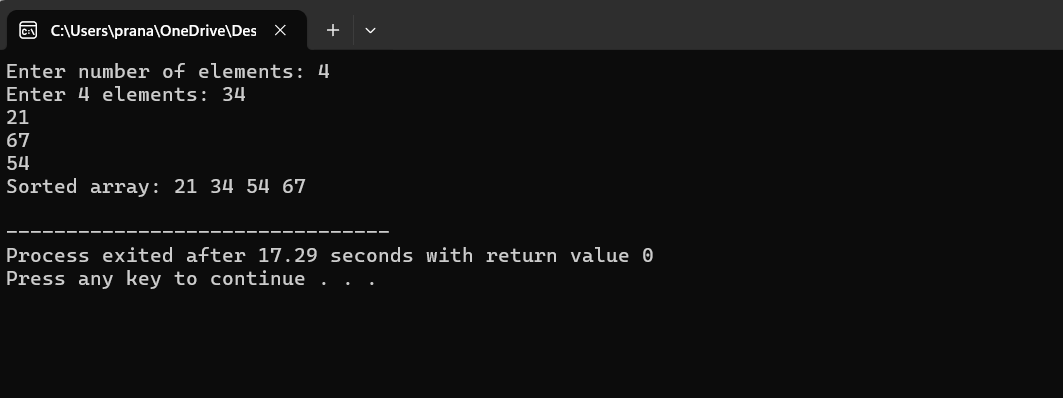
**printf("%d ", arr[i]);**

**}**

**printf("\n");**

**return 0;**

**}**

****

**Q5) WRITE A PROGRAM IN MERGE SORT**

**ANS-**

**#include <stdio.h>**

**#include <stdlib.h>**

**void merge(int arr[], int l, int m, int r) {**

**int i, j, k;**

**int n1 = m - l + 1;**

**int n2 = r - m;**

**int L[n1], R[n2];**

**for (i = 0; i < n1; i++)**

**L[i] = arr[l + i];**

**for (j = 0; j < n2; j++)**

**R[j] = arr[m + 1 + j];**

**i = 0;**

**j = 0;**

**k = l;**

**while (i < n1 && j < n2) {**

**if (L[i] <= R[j]) {**

**arr[k] = L[i];**

**i++;**

**} else {**

**arr[k] = R[j];**

**j++;**

**}**

**k++;**

**}**

**while (i < n1) {**

**arr[k] = L[i];**

**i++;**

**k++;**

**}**

**while (j < n2) {**

**arr[k] = R[j];**

**j++;**

**k++;**

**}**

**}**

**void mergeSort(int arr[], int l, int r) {**

**if (l < r) {**

**int m = (l+r) / 2;**

**mergeSort(arr, l, m);**

**mergeSort(arr, m + 1, r);**

**merge(arr, l, m, r);**

**}**

**}**

**int main() {**

**int n, i;**

**printf("Enter number of elements: ");**

**scanf("%d", &n);**

**int arr[n];**

**printf("Enter %d elements: ", n);**

**for (i = 0; i < n; i++) {**

**scanf("%d", &arr[i]);**

**}**

**mergeSort(arr, 0, n - 1);**

**printf("Sorted array: ");**

**for (i = 0; i < n; i++) {**

**printf("%d ", arr[i]);**

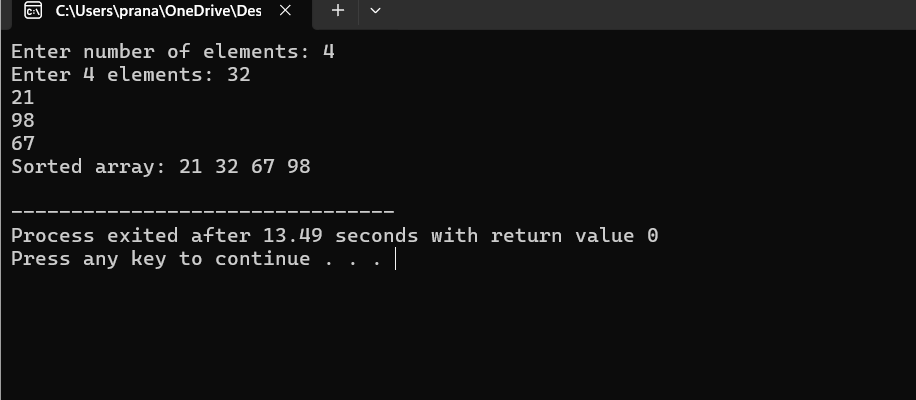
**}**

**printf("\n");**

**return 0;**

**}**

**Output-**

****

**Q6) WRITE A PROGRAM IN QUICK SORT**

**Ans:**

**#include <stdio.h>**

**#include <stdlib.h>**

**int a[5] = {90, 67, 3, 23, 7};**

**void interchange(int i, int j)**

**{**

**int temp = a[i];**

**a[i] = a[j];**

**a[j] = temp;**

**}**

**int partition(int a[], int m, int h)**

**{**

**int v = a[m];**

**int i = m + 1;**

**int j = h;**

**while (i <= j)**

**{**

**while (i <= h && a[i] <= v)**

**{**

**i = i + 1;**

**}**

**while (j >= m && a[j] > v)**

**{**

**j = j - 1;**

**}**

**if (i < j)**

**{**

**interchange(i, j);**

**}**

**}**

**interchange(m, j);**

**return j;**

**}**

**void quick\_sort(int p, int q)**

**{**

**if (p < q)**

**{**

**int j = partition(a, p, q);**

**quick\_sort(p, j - 1);**

**quick\_sort(j + 1, q);**

**}**

**}**

**int main()**

**{**

**printf(" Array before sorting ");**

**for(int i=0;i<5;i++){**

**printf("%d ",a[i]);**

**}**

**quick\_sort(0, 4);**

**printf(" \nArray after sorting :");**

**for (int i = 0; i < 5; i++)**

**{**

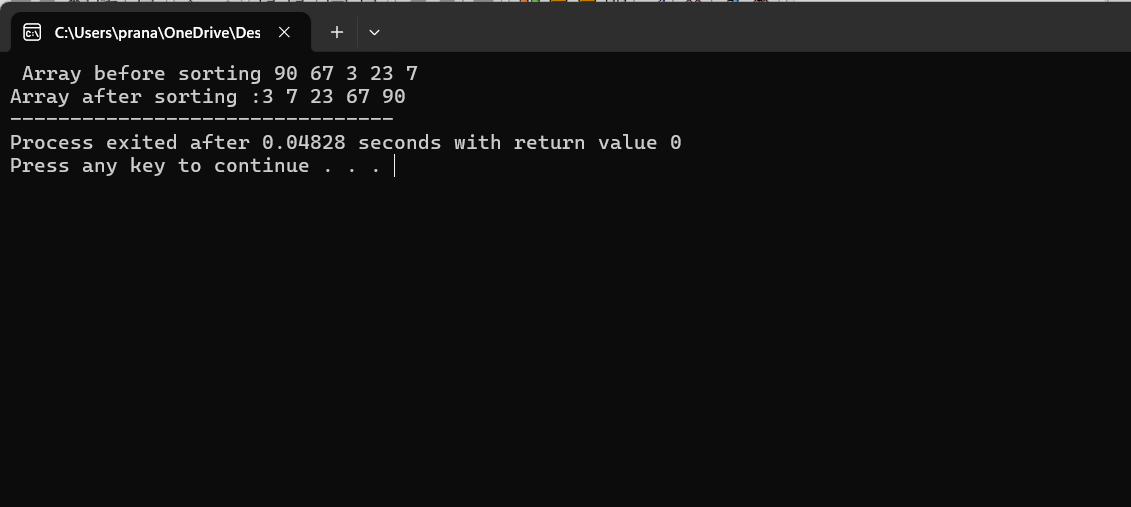
**printf("%d ", a[i]);**

**}**

**return 0;**

**}**

**OUTPUT:**

****

**Q7) WRITE A PROGRAM NON- DETERMINISTIC LINEAR SEARCH**

**ANS-**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**void linear\_search(int a[],int n,int x){**

**srand(time(NULL));**

**int lower = 0, upper = n-1;**

**int rn = (rand() % (upper - lower + 1)) + lower;**

**printf(" randomnly selected number is :%d \n",a[rn]);**

**if(rn==x){**

**printf(" The given number is present in the array st index :%d ",rn);**

**}**

**else{**

**printf(" The given number is not present in the given array ");**

**}**

**}**

**int main(){**

**int a[10]={23,56,43,78,45,21,32,67,87,3};**

**int x ;**

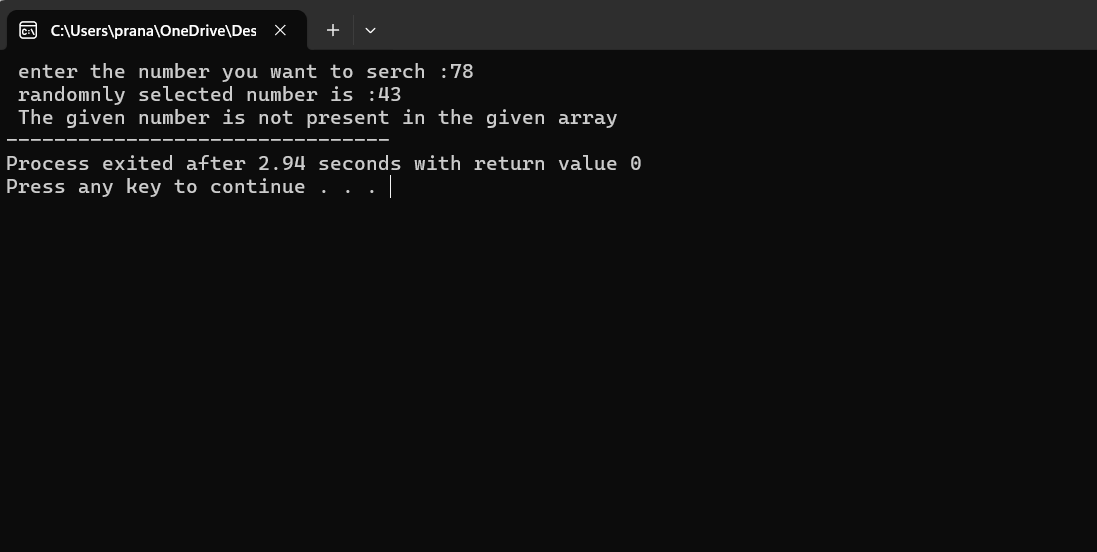
**printf(" enter the number you want to serch :");**

**scanf("%d",&x);**

**linear\_search(a,10,x);**

**}**

**Output:**

****

**Q8) WRITE A PROGRAM FRACTIONAL KNAPSACK**

**ANS-**

**#include <stdio.h>**

**void fractionalKnapsack(int W[], int V[], int M, int n) {**

**float cost[n], total = 0.0;**

**int i, j;**

**for (i = 0; i < n; i++) {**

**cost[i] = (float)V[i] / W[i];**

**}**

**for (i = 0; i < n - 1; i++) {**

**for (j = 0; j < n - i - 1; j++) {**

**if (cost[j] < cost[j + 1]) {**

**float temp = cost[j];**

**cost[j] = cost[j + 1];**

**cost[j + 1] = temp;**

**int tempW = W[j];**

**W[j] = W[j + 1];**

**W[j + 1] = tempW;**

**int tempV = V[j];**

**V[j] = V[j + 1];**

**V[j + 1] = tempV;**

**}**

**}**

**}**

**i = 0;**

**while (i < n) {**

**if (W[i] <= M) {**

**M -= W[i];**

**total += V[i];**

**} else {**

**total += (float)V[i] \* M / W[i];**

**break;**

**}**

**i++;**

**}**

**printf("Maximum Profit: %.2f\n", total);**

**}**

**int main() {**

**int n, M;**

**printf("Enter the number of items: ");**

**scanf("%d", &n);**

**printf("Enter the capacity of the knapsack: ");**

**scanf("%d", &M);**

**int W[n], V[n];**

**printf("Enter the weights and values of each item:\n");**

**for (int i = 0; i < n; i++) {**

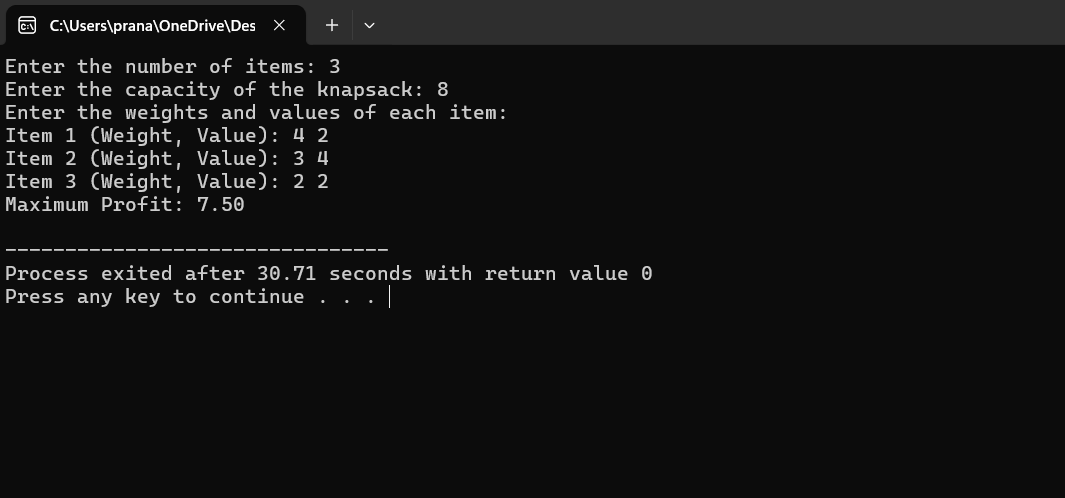
**printf("Item %d (Weight, Value): ", i + 1);**

**scanf("%d %d", &W[i], &V[i]);**

**}**

**fractionalKnapsack(W, V, M, n);**

**}**

**Output:  
**

**Q9) WRITE A PROGRAM FRACTIONAL KNAPSACK NON DETRMINISTIC**

**ANS-**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**void Ndknap(int n, float p[], float w[], float maxcap, float minprof) {**

**int x[20],i;**

**float totalWeight = 0, totalProfit = 0;**

**// Initialize the random number generator**

**srand(time(0));**

**// Select random 0 or 1 for each item**

**for ( i = 0; i < n; i++) {**

**x[i] = rand() % 2;**

**totalWeight += w[i] \* x[i];**

**totalProfit += p[i] \* x[i];**

**}**

**// Check the conditions for failure**

**if (totalWeight > maxcap || totalProfit < minprof) {**

**printf("Failure\n");**

**} else {**

**printf("Success\n");**

**// Print the result vector**

**for ( i = 0; i < n; i++) {**

**printf("%d\t", x[i]);**

**}**

**printf("\n");**

**printf("Total profit is: %f",totalProfit);**

**}**

**}**

**int main() {**

**// Example inputs**

**int n = 5;**

**float p[] = {10, 20, 30, 40, 50};**

**float w[] = {1, 2, 3, 8, 7};**

**float maxcap = 10;**

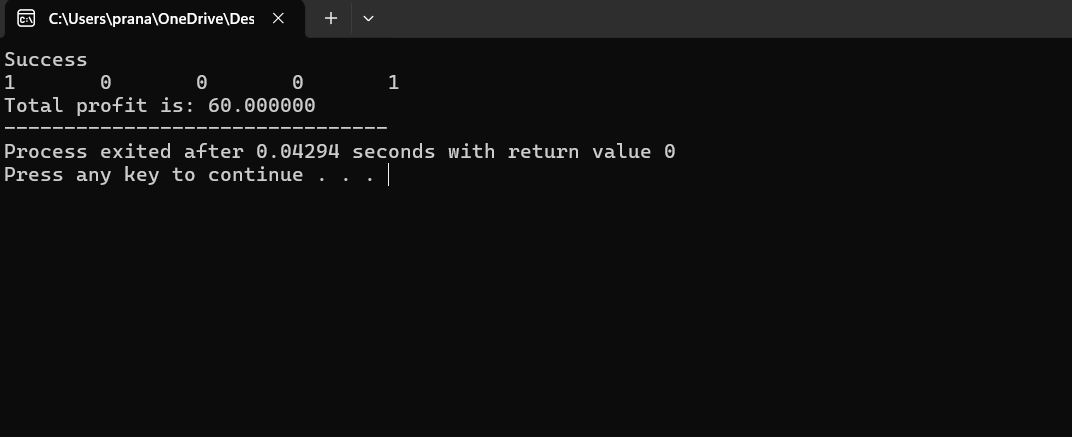
**float minprof = 60;**

**// Call the Ndknap function**

**Ndknap(n, p, w, maxcap, minprof);**

**}**

**OUTPUT;**

****

**Q10) WRITE A PROGRAM NON- DETERMINITIC PRIMALITY TEST**

**ANS-**

**#include <stdio.h>**

**#include<stdlib.h>**

**#include<time.h>**

**int rptesting(int N,int K){**

**int i=1;**

**int r,remainder;**

**L:**

**r = (rand()%(N-2)) +2;**

**remainder = N %r;**

**if(remainder==0)**

**goto out;**

**else**

**i= i+1;**

**if(i<=K)**

**goto L;**

**printf("\nN is prime.\n");**

**return N;**

**out:**

**printf("\nN is composite no.\n");**

**return 0;**

**}**

**int main(){**

**srand(time(0));**

**int N = 10;**

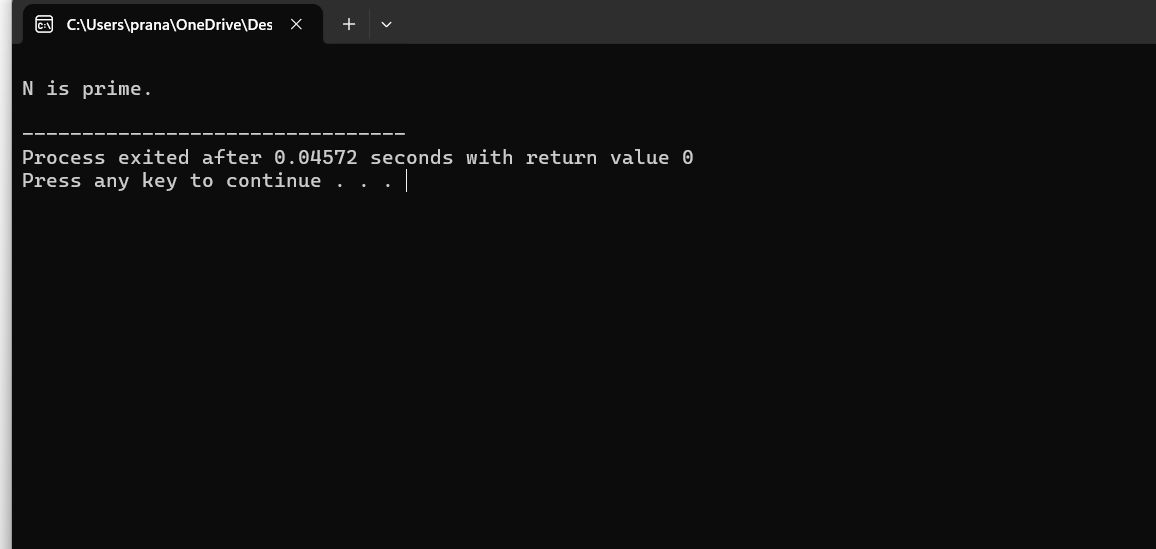
**int K = 5;**

**rptesting(N,K);**

**return 0;**

**}**

**OUTPUT:**

****

**Q11) WRITE A PROGRAM COUNTING SORT**

**ANS;**

**#include <stdio.h>**

**#include <stdlib.h>**

**void countMerge(int a[],int k,int n){**

**int c[k+1],b[n],i;**

**for(i=0;i<=k;i++){**

**c[i]=0;**

**}**

**for(i=0;i<n;i++){**

**c[a[i]]=c[a[i]]+1;**

**}**

**for(i=1;i<=k;i++){**

**c[i]=c[i]+c[i-1];**

**}**

**for(i=n-1;i>=0;i--){**

**b[c[a[i]]-1]=a[i];**

**c[a[i]]=c[a[i]]-1;**

**}**

**for(i=0;i<n;i++){**

**a[i]=b[i];**

**}**

**}**

**int main(){**

**int a[]={5,2,6,3,1,0,4,3,1,1};**

**int k=6,i;**

**int n=sizeof(a)/sizeof(a[0]);**

**countMerge(a,k,n);**

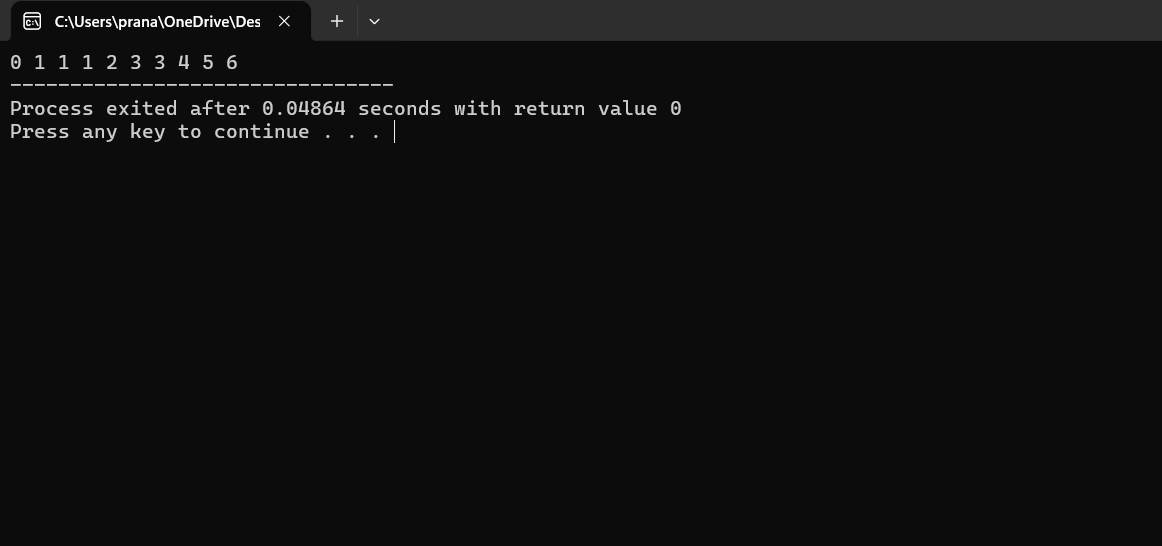
**for( i=0;i<n;i++){**

**printf("%d ",a[i]);**

**}**

**}**

**OUTPUT:**

****

**Q12) WRITE A PROGRAM SUM- SUBSET PROBLEM**

**ANS-**

**#include<stdio.h>**

**#include<stdlib.h>**

**int w[6],x[6]={0},m,r=0,n=6;**

**void print(int x[],int n)**

**{**

**int i;**

**printf("[");**

**for(i=0;i<n;i++)**

**printf("%d ",x[i]);**

**printf("]\n");**

**printf("Subsets are [");**

**for(int i=0;i<6;i++)**

**{**

**if(x[i]==1)**

**{**

**printf("%d ",w[i]);**

**}**

**}**

**printf("]\n");**

**}**

**void sumofsubset(int s,int k,int r)**

**{**

**int i;**

**if(k>=n)**

**return ;**

**x[k]=1;**

**if(s+w[k]==m)**

**{**

**print(x,n);**

**}**

**else if( k+1<n && s+w[k]+w[k+1]<=m)**

**sumofsubset(s+w[k],k+1,r-w[k]);**

**x[k]=0;**

**if(s+r-w[k]>=m && k+1 < n && s+w[k+1]<=m)**

**{**

**sumofsubset(s,k+1,r-w[k]);**

**}**

**}**

**int main()**

**{**

**printf("Enter weights of elements: ");**

**for(int i=0;i<6;i++)**

**{**

**scanf("%d",&w[i]);**

**r+=w[i];**

**}**

**printf("Enter the sum ");**

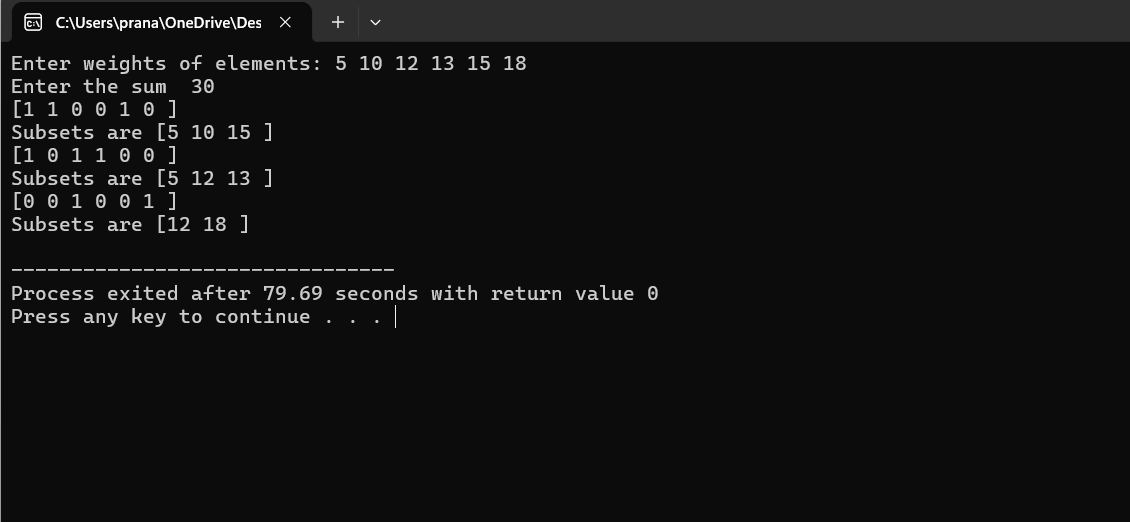
**scanf("%d",&m);**

**//print(w,n);**

**sumofsubset(0,0,r);**

**}**

**OUTPUT-**

****

**Q13)**

**WRITE A PRORAM OF N QUEEN PROBLEMS**

**ANS\-**

**#include <stdio.h>**

**#include <stdbool.h>**

**#define SIZE 8**

**bool isSafe(int board[SIZE][SIZE], int row, int col) {**

**int i, j;**

**for (i = 0; i < col; i++) {**

**if (board[row][i]) {**

**return false;**

**}**

**}**

**for (i = row, j = col; i >= 0 && j >= 0; i--, j--) {**

**if (board[i][j]) {**

**return false;**

**}**

**}**

**for (i = row, j = col; i < SIZE && j >= 0; i++, j--) {**

**if (board[i][j]) {**

**return false;**

**}**

**}**

**return true;**

**}**

**bool solve8QueensUtil(int board[SIZE][SIZE], int col) {**

**if (col >= SIZE) {**

**return true;**

**}**

**for (int i = 0; i < SIZE; i++) {**

**if (isSafe(board, i, col)) {**

**board[i][col] = 1;**

**if (solve8QueensUtil(board, col + 1)) {**

**return true;**

**}**

**board[i][col] = 0;**

**}**

**}**

**return false;**

**}**

**void solve8Queens() {**

**int board[SIZE][SIZE] = {0};**

**if (solve8QueensUtil(board, 0)) {**

**printf("Solution for the 8-Queens problem:\n");**

**for (int i = 0; i < SIZE; i++) {**

**for (int j = 0; j < SIZE; j++) {**

**printf("%c ", board[i][j] ? 'Q' : '.');**

**}**

**printf("\n");**

**}**

**} else {**

**printf("No solution exists for the 8-Queens problem.\n");**

**}**

**}**

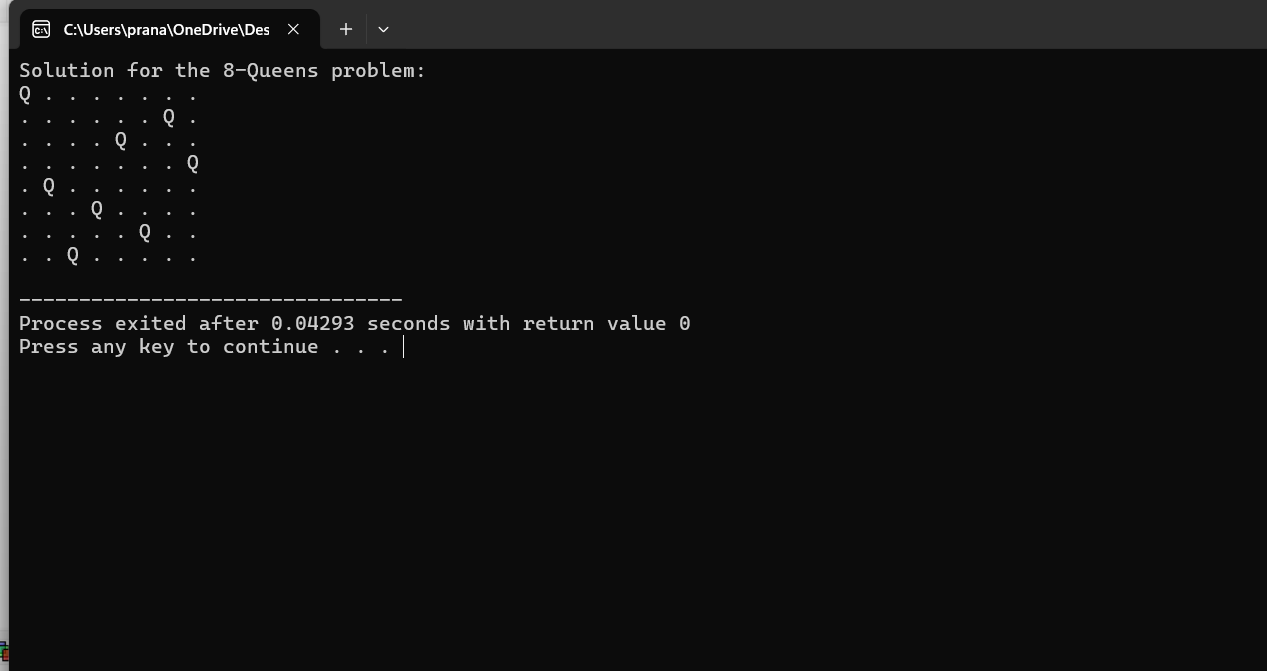
**int main() {**

**solve8Queens();**

**return 0;**

**}**

**OUTPUT-**

****

**Q14) WRITE A PRORAM OF RANDOMISSED QUICK SHORT PROBLEMS**

**ANS-**

**Objective:- randamissed quick short**

**Code:-**

**#include <stdio.h>**

**# include<stdlib.h>**

**#define MAX 20000000**

**int a[MAX];**

**void rQuickSort(int a[], int l, int h){**

**if (l < h){**

**int i = l + rand() % (h - l + 1);**

**int temp = a[l];**

**a[l] = a[i];**

**a[i] = temp;**

**int q = rPartition(a, l, h);**

**rQuickSort(a, l, q);**

**rQuickSort(a, q + 1, h);**

**}**

**}**

**int rPartition(int a[], int l, int h){**

**int pivot = a[l];**

**int i = l - 1;**

**int j = h + 1;**

**while (1){**

**do{**

**j--;**

**} while (a[j] > pivot && j > l);**

**do{**

**i++;**

**} while (a[i] < pivot && i < h);**

**if (i < j){**

**int temp = a[i];**

**a[i] = a[j];**

**a[j] = temp;**

**} else {**

**return j;**

**}**

**}**

**}**

**int main(){**

**FILE \*file = fopen("file2.txt", "r");**

**if (file == NULL){**

**printf("Error opening file\n");**

**return 1;**

**}**

**int n=100;**

**//fscanf(file, "%d", &n);**

**for (int i = 0; i < n; i++){**

**fscanf(file, "%d", &a[i]);**

**}**

**fclose(file);**

**rQuickSort(a, 0, n - 1);**

**printf("Sorted array: ");**

**for (int i = 0; i < n; i++){**

**printf("%d ", a[i]);**

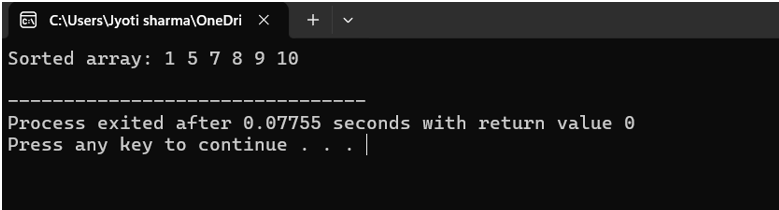
**}**

**printf("\n%d Elements are sorted \n", n);**

**return 0;**

**}**

**OUTPUT;**

****

**Q15) WRITE A PRORAM OF** Comparison between different sorting **PROBLEMS**

**ANS-**

#include<stdio.h>

#include<stdlib.h>

#include "sortingmethod.h"

int main(){

int\* a;

int n=100000;

a=(int\*)calloc(n, sizeof(int));

FILE \*data = fopen("file2.txt", "r");

if(data == NULL)

{

printf("There was a mistake\n");

return 1;

}

int count = 0;

for (; count < n; ++count)

{

if(EOF == fscanf(data, "%d", &(a[count])))

{

break;

}

}

fclose(data);

printf(" which sorting method you want to apply select from list :\n");

printf(" 1.Bubble Sort\n 2.Merge Sort\n 3.Quick Sort ");

int c;

printf(" Select method :");

scanf("%d",&c);

if(c==1){

bubble\_sort(a,n);

}

else if(c==2){

mergesort(a,0,n-1);

}

else if(c==3){

quick\_sort(a,0,n-1);

}

else{

printf(" enter a valid choice");

}

printf(" Sorted Array ");

for(int i=0;i<n;i++){

printf("%d ", a[i]);

}

printf("\n Sucessful ");

}

OUTPUT;  
