**Assignment#01**

**Q. 1. Write a program to generate the n unique random numbers by two different algorithms.**

(Method-1)

#include <stdioh>

#include <stdlib.h>

#include <time.h>

#define randrange(N) rand() / (RAND\_MAX/(N) + 1)//If not to use floating point

#define MAX 100000

static int arr[10];

long long int cad[MAX];

int main (void) {

clock\_t begin = clock();

int i;

srand(time(NULL));

for (i=0; i<MAX; i++) cad[i] = i;

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

int c = randrange(MAX-i);

int t = cad[i];

cad[i] = cad[i+c];

cad[i+c] = t; }

for (i=0; i<10; i++) arr[i] = cad[i] + 1;

for (i=0; i<10; i++) printf("%i\n", arr[i]);

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent);

return 0;

}

Output-

19416 5201 22998 23029 8012 5926 19770 12953 2665 20278

(Method 2)

#include<stdio.h>

#include <time.h>

int main(){

clock\_t begin = clock();

srand(time(NULL));

int vektor[10],random,uniqueflag,I,j;

for(i = 0; i < 10; i++) {

do {

uniqueflag = 1;

random = rand() % 100+ 1;

for (j = 0; j < i && uniqueflag == 1; j++) {

if (vektor[j] == random) uniqueflag = 0; }

} while (uniqueflag != 1);

vektor[i] = random; }

for(i=0;i<10;i++) printf("%d\n",vektor[i]);

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent);

}

Output-

26 90 56 81 53 12 35 67 62 27

**Q. 2. There is an array A of integers of size n which cannot directly be accessed. However, you can get true or false response to queries of the form A[i] < A[j]. It is given that A has only one duplicate pair, and rest all the elements are distinct. So it has n-1 distinct elements and 1 element which is same as one of the n-1 elements. Your task is to identify the indices of the**

**two identical elements in A.**

#include<stdio.h>

int arr[5]={1,6,2,4,4};

int pair\_check(int i,int j){ return arr[i]<arr[j]; }

int main(){

int n=5,i,j,check=0,flag=0;

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

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

if((pair\_check(i,j) || pair\_check(j,i))==0){

printf("%d\t%d",i,j);

flag=1;

break;

}

}

if(flag==1) break;

}

}

Input-

arr[5]={1,6,2,4,4}

Output-

3 4

**Q. 3. Write a function customSort takes a "num" array of integers as an input along with the array of "weights" which contains the weights of the corresponding integers. We wish to sort this vector "num" from the indices start to end based on the fact that numbers with higher weights appear first and in case the weights are equal, we put the greater number first in the list. We also wish efficient algorithms. We expect an O (n\*log(n)) algorithm here instead of a O(n^2) one.**

#include<stdio.h>

#include<time.h>

int indx[];

void check(int a[],int arr[],int n){

int temp[n]; int i,j,k=0;

for(i=0;i<n;i++) temp[i]=arr[indx[i]];

for(i=k,j=i+1;i<n-1 && j<n;i++,j++){

if(a[i]==a[j]){

k=k+2;

if(temp[i]>temp[j]){ int t=temp[i]; temp[i]=temp[j]; temp[j]=t; }

}}

printf("Sorted array: \n");

for(i=0;i<n;i++) printf("%d ",temp[i]);

}

void swap(int\* a, int\* b){ int t = \*a; \*a = \*b; \*b = t; }

int partition (int arr[], int low, int high,int index[]){

int pivot = arr[high], i = (low - 1);

for (j = low; j <= high- 1; j++){

if (arr[j] <= pivot) {

i++;

swap(&arr[i], &arr[j]);

swap(&index[i],&index[j]); } }

swap(&arr[i + 1], &arr[high]);

swap(&index[i + 1], &index[high]);

return (i + 1); }

void custom(int arr[], int low, int high){

if (low < high){

int pi = partition(arr, low, high,indx);

custom(arr, low, pi - 1);

custom(arr, pi + 1, high); } }

int main(){

printf("Enter size of array\t");

int weight[n],arr[n],n,i,j;

scanf("%d",&n);

printf("Enter Original array\n");

for(i=0;i<n;i++) scanf("%d",&arr[i]);

printf("Enter weights of elements of Original array\n");

for(i=0;i<n;i++) scanf("%d",&weight[i]);

clock\_t begin = clock();

indx[n];

for(i=0;i<n;i++) indx[i]=i;

custom(weight, 0, n-1);

check(weight,arr,n);

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent);

return 0;

}

Output-

Enter size of array 5

Enter Original array

1 2 3 4 5

Enter weights of elements of Original array

9 7 9 6 11

Sorted array:

4 2 1 3 5

**Q. 4. Write a program to find the value of xn, where ‘x’ and ‘n’ are the real numbers using 3 different algorithms.**

#include<stdio.h>

#include<math.h>

double Method3(double,int,int);

double power(double,int);

int main(){

double ans,x,n; int p,q;

printf("Enter x "); scanf("%lf",&x);

printf("Enter n as p/q "); scanf("%d",&p); scanf("%d",&q);

n=p\*1.0/q; printf("\n Method 3:\n");

ans=Method3(x,p,q);

printf("%lf^%lf=%lf",x,n,ans); }

double Method3(double x,int p,int q){

double ans1,ans2,c; int a,b;

c=power(x,p); ans2=10;

do{ ans1=ans2;

ans2=ans1-((power(ans1,q)-c)/(q\*power(ans1,q-1)));

a=ans1; b=ans2;

}while(a!=b);

return ans2; }

double power(double x,int n){

int i; double ans=1.00;

for(i=0;i<n;i++) ans=ans\*x;

return ans; }

Output-

Enter x 2

Enter n as p/q 1 2

Method 3:

2.000000^0.500000=1.444238

**Q. 5. Write a program to find the factorial of the given number (0 < n < 10,000,000,000).**

#include<stdio.h>

#include<time.h>

#define MAX 100000

unsigned int multiply(int x, unsigned int res[], int res\_size){

int carry=0,i;

for (i=0; i<res\_size; i++){

int prod=res[i]\*x + carry;

res[i]=prod % 10;

carry=prod/10; }

while (carry){

res[res\_size] = carry%10;

carry=carry/10;

res\_size++; }

return res\_size; }

void factorial(int n){

unsigned int res[MAX];

res[0]=1;

int res\_size=1,x,i;

for (x=2; x<=n; x++) res\_size = multiply(x, res, res\_size);

printf("Factorial of given number is \n");

for (i=res\_size-1; i>=0; i--) printf("%u ",res[i]); printf("\n");

}

int main(){

int a;

printf("Enter value\n"); scanf("%d",&a);

clock\_t begin = clock();

factorial(a);

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent);

}

Output-

Enter value

10

Factorial of given number is

3 6 2 8 8 0 0

**Q. 6. Write a program to find the nth Fibonacci number with two different approach and compare their time complexity.**

#include <stdio.h>

#include<time.h>

int fibo\_rec(int num) {

if(num<=1) return 1;

else return fibo\_rec(num-1) + fibo\_rec(num-2);

}

int fibo\_it(int num) {

int i,result=1,prev=0,t=0;

if(num==0 || num==1) return 1;

else{

for(i=2;i<=num;i++) {

t=prev;

prev = result;

result=t+result;

}

return result;

}}

int main() {

printf("Enter number\n");

int num;

scanf("%d",&num);

int result=0;

printf("Enter choice:\n1. Iteration\n2. Recursion\n");

int ch;

scanf("%d",&ch);

if(ch==1){

clock\_t begin = clock();

result = fibo\_it(num);

printf("%d\n",result);

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent);

}

if(ch==2){

clock\_t begin1 = clock();

result = fibo\_rec(num-1);

printf("%d\n",result);

clock\_t end1 = clock();

double time\_spent1 = (double)(end1 - begin1) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent1);

}

}

**Q. 7. An array of integers is said to be a straight-K, if it contains K elements that are K consecutive numbers. For example, the array {6, 1, 9, 5, 7, 15, 8} is a straight because it contains 5, 6, 7, 8, and 9 for K=5. Write a program to finds the maximum value of K for the given number of integers.**

#include<stdio.h>

#include <time.h>

void swap(int\* a, int\* b){

int t = \*a; \*a = \*b; \*b = t; }

int partition (int arr[], int low, int high){

int pivot = arr[high];

int i = (low - 1),j;

for (j = low; j <= high- 1; j++){

if (arr[j] <= pivot){ i++; swap(&arr[i], &arr[j]); }

}

swap(&arr[i + 1], &arr[high]);

return (i + 1);

}

void quickSort(int arr[], int low, int high){

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}}

int main(){

printf("Enter size of array\t");

int n,i,j;

scanf("%d",&n);

int arr[n];

printf("Enter Original array\n");

for(i=0;i<n;i++) scanf("%d",&arr[i]);

clock\_t begin = clock();

quickSort(arr, 0, n-1);

int counter[n];

for(i=0;i<n;i++) counter[i]=0;

int count=1;

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

if(arr[i]+1==arr[i+1]) count++;

else{ counter[i]=count; count=1; }

}

int max=0;

for(i=0;i<n;i++){ if(counter[i]>max) max=counter[i]; }

printf("%d",max);

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent);

return 0; }

Output- Enter size of array 10

Enter Original array 1 2 3 6 7 8 9 10 11 0

6

**Q. 8. You have an array like :-{ 5,5,5,13,6,13,6,7,8,7,-1) Write and analyze an algorithm to arrange this array in sorted form based upon the number of occurrence e.g.: above should look like this after execution of your also {-1,5,5,5,13,13,6,6,7,7,8) 13 comes before 6 because it has same number of occurrence as 6 but it come first in the parent array.**

#include<stdio.h>

#include<time.h>

int main(){

int flag,x,arr[100],n,i,j,c=0,num,k;

for(i=0;i<100;i++) arr[i]=0;

printf("Enter array size:\n"); scanf("%d",&n);

n\*=2;

printf("Enter Elements:\n");

for(i=0;i<n;i+=2){

flag=0;

scanf("%d",&num);

for(j=0;j<=i;j+=2){

if(arr[j]==num){ arr[j+1]++; flag=1; }

}

if(flag==0){

c++;

arr[c\*2-2]=num;

arr[c\*2-1]++;

}

}

clock\_t begin = clock();

int stack[200];

for(i=0;i<200;i++) stack[i]=0;

int top=-1;

for(k=1;k<n/2;k++){

for(j=c\*2-1;j>0;j-=2){

if(arr[j]==k) stack[++top]=j;

}

}

printf("\nSorted array is:\n");

for(x=top;x>-1;x--){

for(j=arr[stack[x]];j>0;j--)

printf("%d ",arr[stack[x]-1]);

}

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent);

}

Output-

Enter array size: 10

Enter Elements: 5 7 8 7 5 12 12 12 5 1

Sorted array is:

5 5 5 12 12 12 7 7 8 1

**Q. 9. Write a program to find the determinant of the given matrix of size NxN using recursion.**

#include<stdio.h>

#include<math.h>

#define N 5

#include <time.h>

int determinant(int[N][N],int);

void init(int[N][N],int);

int main(){

int result ,a[N][N],l,n;

printf("Enter matrix(max 5x5):"); scanf("%d %d",&l,&n);

if(l!=n) printf("l!=n Not square matrix!!\n");

else{

init(a,n);

clock\_t begin = clock();

result = determinant(a,n);

printf("\nDeterminant: %d",result);

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent);

}}

void init(int b[N][N],int m){

printf("Enter elements:\n");

int i,j;

for(i=0;i<m;i++)

for(j=0;j<m;j++) scanf("%d",&b[i][j]);

printf("Matrix is:\n");

for(i=0;i<m;i++){

for(j=0;j<m;j++) printf("%d ",b[i][j]);

printf("\n"); }

}

int determinant(int b[N][N],int m){

int p,i,j,result = 0,c[N][N];

if(m==1) return b[0][0];

else if(m==2){

result = b[0][0]\*b[1][1] - b[0][1]\*b[1][0];

return result; }

for(p=0;p<m;p++){

int h = 0,k = 0;

for(i=1;i<m;i++){

for( j=0;j<m;j++){

if(j==p) continue;

c[h][k] = b[i][j]; k++;

if(k == m-1){ h++; k = 0; }

}}

result = result + b[0][p]\*pow(-1,p)\*determinant(c,m-1);

}

return result;

}

Output-

Enter matrix(max 5x5):3

3

Enter elements:

1

2

3

4

5

6

7

8

9

Matrix is:

1 2 3

4 5 6

7 8 9

Determinant: 0

**Q. 10. Write a program to display the elements of given matrix of size NxN in clockwise spiral order using recursion.**

#include <stdio.h>

#include <stdlib.h>

#include<time.h>

void spiralscan(int a[][100],int n,int c,int count){

int i,j;

if(count>=n\*n) return;

for(j=c;j<n-c;j++){ printf("%d ",a[c][j]); count++; }

for(i=c+1;i<n-c;i++){ printf("%d ",a[i][n-c-1]); count++; }

for(j=n-2-c;j>=c;j--){ printf("%d ",a[n-1-c][j]); count++; }

for(i=n-2-c;i>c;i--){ printf("%d ",a[i][c]); count++; }

spiralscan(a,n,c+1,count);

}

int main(){

int n,m,i,j,a[100][100];

printf("Enter rows and columns: "); scanf("%d%d",&n,&m);

if(n!=m) printf("Not a NxN matrix!\n");

else{

printf("\nEnter matrix: \n");

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

for(j=0;j<n;j++) scanf("%d",&a[i][j]); }

printf("\nSpiral Order:\n");

clock\_t begin = clock();

spiralscan(a,n,0,0);

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("----------------%lf------------------",time\_spent);

}

return 0;

}

Output-

Enter rows and columns: 3 3

Enter matrix:

1

2

3

8

9

4

7

6

5

Spiral Order:

1 2 3 4 5 6 7 8 9