**A logo of a book and a candle

Description automatically generatedKathmandu**

**Bernhardt College**

Bafal (Ringroad), Kalanki

*(Affiliated to TU)*

Lab Report No:

Lab report on Design And Analysis of Algorithm

Submission Date:2080-06-11

Submitted by: Prajwol Subedi Submitted To:

Symbol No: 26094/077 CSIT Department

Section: A

Signature

1. **Program to calculate the Greatest Common Divisor (GCD) of two numbers.**

#include<stdio.h>

#include<conio.h>

int main() {

int a, b, q, r;

printf("Enter the values of a and b: ");

scanf("%d%d", &a, &b);

int r1 = a;

int r2 = b;

while(r2 > 0){

q = r1 / r2;

r = r1 - q \* r2;

r1 = r2;

r2 = r;

}

printf("GCD(%d, %d) = %d", a, b, r1);

return 0;

}

**OUTPUT:**

**A screenshot of a video game

Description automatically generated**

1. **Program to display the Fibonacci sequence.**

#include<stdio.h>

#include<conio.h>

int main(){

int a = 0, b = 1, c;

int n;

printf("Enter the no. terms in the sequence: ");

scanf("%d", &n);

printf("Fibonacci Series is: \n");

printf("%d %d ", a, b);

while(n - 2 > 0){

c = a + b;

a = b;

b = c;

printf("%d ", c);

n = n - 1;

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

1. **Program to calculate the factorial of given number.**

#include<stdio.h>

#include<conio.h>

int main(){

int n, i, fact = 1;

printf("Enter number whose factorial is to be calculated: ");

scanf("%d", &n);

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

fact = fact \* i;

}

printf("Factorial of %d is %d.", n, fact);

return 0;

}

**OUTPUT:**

**A black screen with white text

Description automatically generated**

1. **Program to implement Sequential search technique.**

#include<stdio.h>

#include<conio.h>

int main(){

int i, element, j=0;

int n = 6;

int a[] = {1, 100, 2, 0, 5, 6};

printf("Enter the element to search: ");

scanf("%d", &element);

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

if(a[i] == element){

printf("%d is present at position %d.", element, i+1);

j++;

}

}

if(j == 0){

printf("%d is not present in array.", element);

}

return 0;

}

**OUTPUT:**

**A screen shot of a computer

Description automatically generated**

1. **Program to implement Binary search technique.**

#include<stdio.h>

#include<conio.h>

int main(){

int size, i, n, mid, low, high, found = 0;

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

scanf("%d", &size);

int a[size];

printf("Enter the element of array: ");

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

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

}

printf("Enter the element to search: ");

scanf("%d", &n);

low = 0;

high = size - 1;

while(high >= low){

mid = (high + low) / 2;

if(a[mid] == n){

printf("%d is present at position %d", n, mid+1);

found = 1;

break;

}

else if(a[mid] > n){

high = mid - 1;

}

else{

low = mid + 1;

}

}

if(found == 0){

printf("%d is not present.", n);

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

1. **Program to implement Bubble sort.**

#include<stdio.h>

#include<conio.h>

int main() {

int i, j, temp, n;

printf("Enter size of array: ");

scanf("%d", &n);

int a[n];

printf("Enter elements of array: ");

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

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

}

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

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

if(a[j] > a[j+1]){

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

printf("Sorted Array: ");

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

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

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

1. **Program to implement Selection sort.**

#include<stdio.h>

#include<conio.h>

int main() {

int i, j, temp, n;

printf("Enter size of array: ");

scanf("%d", &n);

int a[n];

printf("Enter elements of array: ");

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

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

}

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

int least = i;

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

if(a[j] < a[least]){

least = j;

}

}

if(least != i){

temp = a[i];

a[i] = a[least];

a[least] = temp;

}

}

printf("Sorted Array: ");

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

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

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

1. **Program to implement Insertion sort.**

#include<stdio.h>

#include<conio.h>

int main() {

int n, i, j, p;

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

scanf("%d", &n);

int a[n];

printf("Enter the elements of array: ");

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

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

}

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

p = a[i];

j = i - 1;

while (j>=0 && a[j]>p){

a[j+1] = a[j];

j--;

}

a[j+1] = p;

}

printf("Sorted array is: ");

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

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

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

1. **Program to implement Quick sort.**

#include<stdio.h>

#include<conio.h>

void quicksort(int [], int, int);

int partition(int [], int, int);

int main(){

int n, i;

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

scanf("%d", &n);

int a[n];

printf("Enter the elements of array: ");

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

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

}

int l = 0;

int r = n-1;

quicksort(a, l, r);

printf("Sorted Array is: ");

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

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

}

return 0;

}

void quicksort(int a[], int l, int r){

int q;

if(l < r){

q = partition(a, l, r);

quicksort(a, l, q-1);

quicksort(a, q+1, r);

}

}

int partition(int a[], int l, int r){

int pivot, i, j, temp1, temp2;

pivot = a[r];

i = l - 1;

for(j=l ; j<=r-1 ; j++){

if(a[j] <= pivot){

i = i + 1;

temp1 = a[i];

a[i] = a[j];

a[j] = temp1;

}

}

temp2 = a[i+1];

a[i+1] = a[r];

a[r] = temp2;

return i+1;

}

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

**10. Program to implement Merge sort.**

#include<stdio.h>

#include<conio.h>

#include<math.h>

void merge\_sort(int [], int, int);

void merge(int [], int, int, int);

int main() {

int n, i;

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

scanf("%d", &n);

int a[n];

printf("Enter the elements of array: ");

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

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

}

int p = 0;

int r = n - 1;

merge\_sort(a, p, r);

printf("Sorted Array is: ");

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

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

}

return 0;

}

void merge\_sort(int a[], int p, int r){

int q;

if(p >= r){

return;

}

q = floor((p + r) / 2);

merge\_sort(a, p, q);

merge\_sort(a, q+1, r);

merge(a, p, q, r);

}

void merge(int a[], int p, int q, int r){

int nl, nr, i, j, k;

nl = q - p + 1;

nr = r - q;

int L[nl], R[nr];

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

L[i] = a[p+i];

}

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

R[j] = a[q+j+1];

}

i = 0, j = 0, k = p;

while(i<nl && j<nr){

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

a[k] = L[i];

i++;

}

else{

a[k] = R[j];

j++;

}

k++;

}

while(i < nl){

a[k] = L[i];

i++;

k++;

}

while(j < nr){

a[k] = R[j];

j++;

k++;

}

}

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

1. **Program to implement Randomized quick sort.**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

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

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int partition(int a[], int l, int r) {

srand(time(NULL));

int k = l + rand() % (r - l + 1);

swap(&a[k], &a[r]);

int pivot = a[r];

int i = l - 1;

int j;

for (j=l ; j<=r-1 ; j++) {

if (a[j] <= pivot) {

i++;

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

}

}

swap(&a[i + 1], &a[r]);

return i + 1;

}

void randomizedQuickSort(int a[], int l, int r) {

if (l < r) {

int q = partition(a, l, r);

randomizedQuickSort(a, l, q - 1);

randomizedQuickSort(a, q + 1, r);

}

}

int main() {

int n, i;

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

scanf("%d", &n);

int a[n];

printf("Enter the elements of array: ");

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

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

}

int l = 0;

int r = n - 1;

randomizedQuickSort(a, l, r);

printf("Sorted array: \n");

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

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

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

1. **Program to implement Heap sort.**

#include<stdio.h>

#include<conio.h>

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

int tmp = \*a;

\*a = \*b;

\*b = tmp;

}

void heapify(int arr[], int n, int i) {

int max = i;

int leftChild = 2 \* i + 1;

int rightChild = 2 \* i + 2;

if (leftChild < n && arr[leftChild] > arr[max])

max = leftChild;

if (rightChild < n && arr[rightChild] > arr[max])

max = rightChild;

if (max != i) {

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

heapify(arr, n, max);

}

}

void heapSort(int arr[], int n) {

int i;

for (i=n/2-1 ; i>=0 ; i--)

heapify(arr, n, i);

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

swap(&arr[0], &arr[i]);

heapify(arr, i, 0);

}

}

int main() {

int n, i;

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

scanf("%d", &n);

int arr[n];

printf("Enter the elements of array: ");

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

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

}

heapSort(arr, n);

printf("Sorted array: \n");

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

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

}

return 0;

}

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**