**1). 5 Student data should be stored in array(ID, Name, Address) and display data value.**

#include<stdio.h>

struct student{

int ID;

char Name[20];

char Address[30];

} stu[30];

int main()

{

printf(“Enter the information of students : \n”);

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

{

printf(“ Enter ID : “);

scanf(“%d”, stu[i].ID);

printf(“Enter Name : “);

scanf(“%s”, stu[i].Name);

printf(“Enter Address : “);

scanf(“%s”, stu[i].Address);

}

printf(“Displaying Information : \n”);

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

{

printf(“ID : %d”, stu[i].ID);

printf(“Name : %s”, stu[i].Name);

puts(stu[i].Name);

printf(“Address : %s”, stu[i].Address);

puts(stu[i].Address);

}

return 0;

}

**1). Algorithm**

1. Start

2. Define a structure student with variables for ID,

Name & Address.

3. Then create a structure variable stu [20] which is an array.

4. Define the main function.

5. Initialize loop: for(i=0; i<5; i++)

6. Next, take user input for storing the value of ID, Name,

& Address.

7. Take another loop to Display the Array Values.

loop: for(i=0; i<5; i++)

8. Stop

**2).** **Sort using Bubble sort based on its Employee ID**

#include<stdio.h>

int main()

{

int a[20],n, i ,j ,k ,b ,temp;

printf(“Enter the size of array: “);

scanf(“%d”, &n);

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

{

printf(“Enter Employee ID: “);

scanf(“%d”, &a[i]);

}

for(b=0; b<n-1 ;b++);

{

for(k=0 ; k<n-1; k++);

{

If(a[k]>a[k+1]);

{

temp = a[k];

a[k] = a[K+1];

a[k+1] = temp;

}

}

}

for(j=0; j<n; j++);

{

printf(“%d ”,a[j]);

}

}

**2. Bubble sort Algorithm**

1. Ask the user to enter the size of an element.

2. Print those elements by using a loop.

3. Take a loop for making a passes.

4. Take another loop for making iteration.

5. Check if the first element is greater or not than

, the second element.

6. If true, then swap first and second element using

One temp variable.

7. Repeat until all the iterations are made.

8. Now finally, Display the information in array

Using a new loop.

**3). Sort using Selection sort based on its Employee ID**

#include<stdio.h>

int main()

{ int a[10], i, j, n, min, temp;

printf(“Enter the size of an array :”);

scanf(“%d”, &n);

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

{

printf(“Enter Employee ID : “);

scanf(“%d ”,&a[i]);

}

for(k=0; k<n; k++)

{

min = k;

for(b=k+1; b<n; b++)

{

If(a[b] < a[min])

{ min = j; }

}

If (min != i)

{

temp = a[min];

a[min] = a[i];

a[i] = temp

}

}

for(j=0; j<n; j++);

{

printf(“%d ”,a[j]);

}

}

**4) Merge Sort Algorithm**

Merge(A, L, nL, R, nR)

{

i🡨j 🡨k🡨0;

while(i< nL && j<nR)

{

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

{ A[k] 🡨 L[i];

i 🡨 i + 1;

}

else

{ A[k] 🡨 R[j];

j 🡨j+1;

}

K 🡨 k + 1;

}

while(i < nL)

{ A[k] 🡨 L[i];

i 🡨 i + 1;

K 🡨k + 1;

while(i<nR)

{ A[k] 🡨 R[j];

J 🡨 j + 1;

K 🡨 K + 1;

}

}

Mergesort(A,n)

{ if(n<2)

{

return;

}

Mid 🡨 n/2;

Left 🡨 Create an Array with size(mid);

Right 🡨 Create an Array with size(n-mid);

for i🡨0 to mid-1

{

Left[i] 🡨 A[i];

}

For i🡨 mid to n-1;

{

Right[i-mid] 🡨 A[i]

}

Mergesort(left, mid)

Mergesort(right, n-mid)

Merge(A, left, mid, right, n-mid)

}

**5). Quicksort Algorithm**

int partition(A, start, end)

{

int pivot 🡨 A[end];

int p\_Index 🡨 start;

for i🡨start to end-1

{

If(A[i] <=pivot)

{

Swap (A[i], A[p\_Index]);

p\_Index 🡨 p\_Index + 1;

}

}

Swap(A[end], A[p\_Index]);

return p\_Index;

}

void Quicksort(A, start, end)

{

p\_Index 🡨 partition(A, start, end);

Quicksort(A, start, p\_Index-1);

Quicksort(p\_Index+1, end);

}

**6). Sort Using Stack based on its Employee Id.**

Import java.util.Stack;

public class SortStack

{

public static void sort(Stack<Integer> st) {

if(st.isEmpty()) {

return;

}

int temp = st.pop();

sort(st);

insertAtCorrectPosition(st, temp);

}

Public static void insertAtCorrectPosition(Stack<Integer> st, int temp)

{

If(st.isEmpty() || st.peek() < temp) {

st.push(temp);

return;

}

int element = st.pop();

insertAtCorrectPosition(st, temp);

st.push(element);

}

public static void main(String[] args) {

Stack<Integer> st = new Stack<>();

st.push(2392);

st.push(2395);

st.push(2391);

st.push(2393);

st.push(2394);

System.out.println(st);

sort(st);

System.out.println(st);

}

}

**Output:**  [2392, 2395, 2391, 2393, 2394]

[2391, 2392, 2393, 2394, 2395]