**ASSIGNMENT - 5**

**1.PROBLEM STATEMENT**

Write a program in C to implement selection sort.

**2. ALGORITHMS**

**Algorithm Selection\_Sort**

**Input:** A pointer to an integer array name arr[1…n] of size n.

**Output:** The same array arr in sorted order.

**Remarks:** Elements are sorted in ascending order and it is assumed that the array is not empty.

**Steps:**

1. **For** i=1 to (n-1) **do** // performing (n-1) iterations
2. j=**Findmin**(i,n) // finding the minimum from the given range
3. **If**(i ≠ j) **then** // if minimum element is not the current element
4. **Swap**(arr[i] ,arr[j]) // exchange the two elements
5. **EndIf**
6. **EndFor**
7. **Stop**

**Algorithm Findmin**

**Input:** A pointer to an array named arr[1…n] of size n with left and right being the given range.

**Output:** The index of the smallest integer in the list.

**Remarks:** If more than one minimum elements are present , the index of the first occurrence is returned.

**Steps:**

1. minele = arr[left] // taking left element as smallest
2. minloc = left // holding the index of left element
3. **For** i= left+1 to right **do** // search the array
4. **If**(arr[i]<minele) **then** // if an element is smaller than minele
5. minele=arr[i] // update minele
6. minloc=I // update minloc
7. **EndIf**
8. **EndFor**
9. **Return** minloc // return the location of the smallest element
10. **Stop**

**Algorithm Swap**

**Input:** The two variables named a and b whose data is to be swapped.

**Output:** The two variables a and b with interchanged data.

**Remarks:** The variables must be passed as pointers.

**Steps:**

1. a = a + b // a holds sum of a and b
2. b = a – b // b holds the previous value of a
3. a = a – b // a holds the previous value of b
4. **Stop**

**3.Source Code**

#include<stdio.h>

#include<stdlib.h>

// function to interchange two elements

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

{

\*a=\*a+\*b;

\*b=\*a-\*b;

\*a=\*a-\*b;

}

// function to display an array

void disparr(int\*arr,int num)

{

int i;

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

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

}

//function to take input in an array

void getarr(int \*arr,int num)

{

int i;

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

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

}

// function to find minimum element in an array

int findmin(int \*arr,int start,int end)

{

int i,minloc;

minloc=start-1; // element at starting location taken as smallest

for(i=start;i<end;i++){ // traversing

if(arr[i]<arr[minloc]) // if element is smaller than assumed one

minloc=i; // update minimum element’s location

}

**return** minloc;

}

void selectionsort(int \*arr,int num)

{

int i,j,minloc;

for(i=0;i<num-1;i++)

{

j=i+1; //setting j at the start of unsorted list

minloc=findmin(arr,j,num); // finding the minimum element

if(i!=minloc) // if minimum element is not the current element

swap(&arr[i],&arr[minloc]); // swap the elements

printf("\n\nPASS %d: ",i+1);

disparr(arr,num);

printf("\n");

}

}

int main(void)

{

int \*arr,num,i;

printf("Enter The Number Of Elements Needed: ");

scanf("%d",&num);

arr = (int\*)calloc(num,sizeof(int)) //creating array in heap

// checking if atleast two elements are present

if(num<2)

{

printf("Invalid Array Length\nPlease Retry");

return 0;

}

printf("\nEnter %d Elements Of The Array: ",num);

getarr(arr,num);

selectionsort(arr,num);

printf("\n\nSorted Array: ");

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

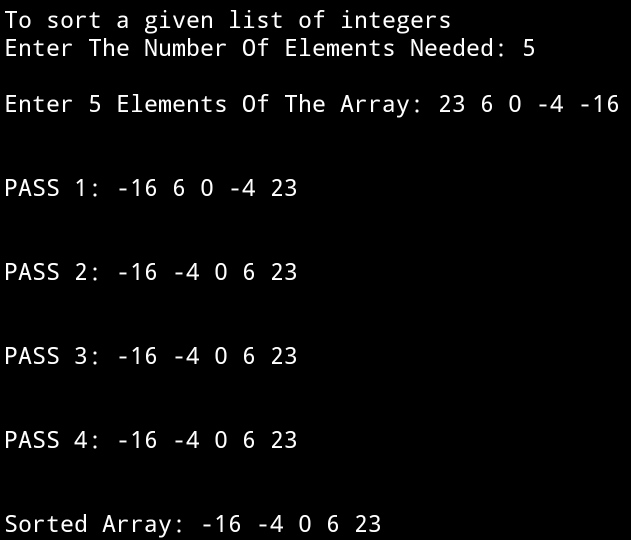
printf("%d ",arr[i]);

return 0;

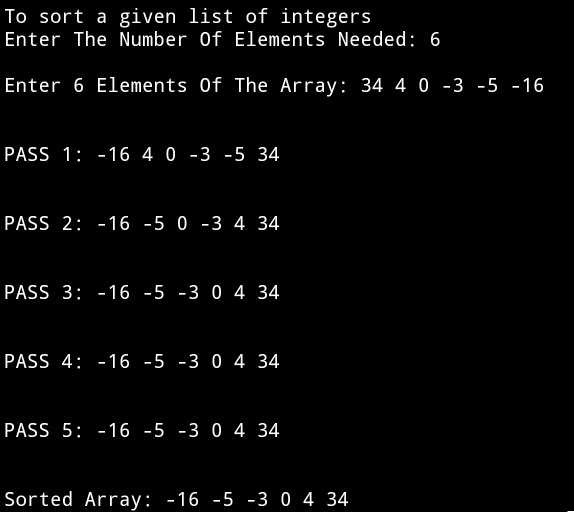
}

**4.OUTPUT**

**SET 1: Odd Sized Input Array**

****

**SET 2: Even Sized Input Array**

****

**5.DISCUSSIONS**

1. **Variable Description**

**In main function**

* **\*arr:** pointer to the array.
* **num:** size of the array.
* **i:** loop counter.

**In selectionsort function**

* **i,j:** loop counter.
* **minloc:** holds the index of the minimum element in given range.

**In findmin function**

* **i:** loop counter.
* **minloc:** holds the index of the minimum element in given range.

**In getarr and disparr functions**

* **i:** loop counter.

1. **Limitations**

* The program uses an integer array to hold the list of integers entered by the user, since arrays are static data structures ,their size cannot be manipulated once it is allocated in the memory.

1. **Uses**

* The above program can be used to sort any list of integers in ascending order. It can be used to sort a list of employees working in an organization in ascending order of their wages.

1. **Future Scope**

* The list of integers can be stored in a linked list , enabling more elasticity in manipulation of size of the list.