CSE2003- Data Structures and Algorithms

LAB ASSIGNMENT-4

Slot: L53 + L54

Faculty: GAYATHRI P Mam

Date: 24-09-2020

19BCE0829 KSHEERAJ KANDRA

QUESTION-1:

1. Implement C program to perform sorting of n numbers using heap sort technique.

PSEUDO CODE:

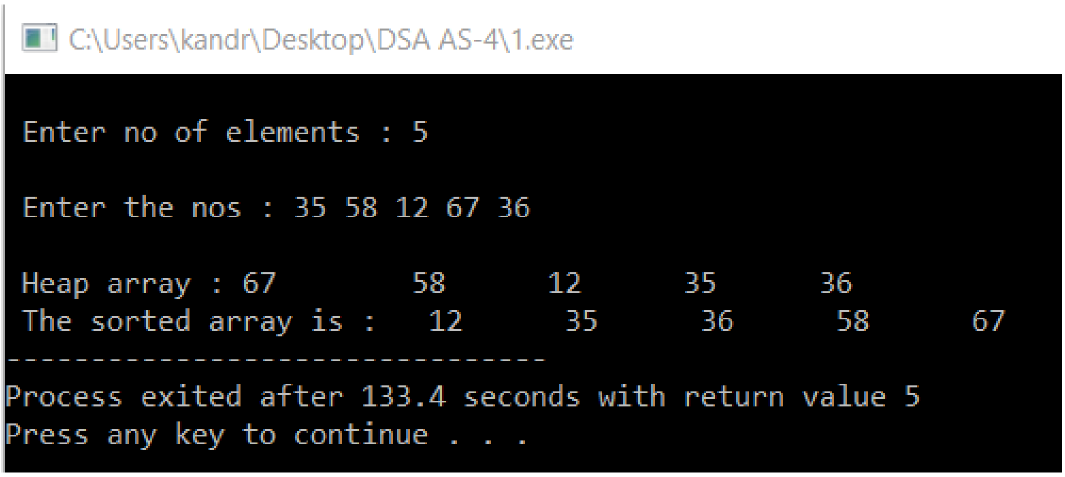
```
void main(){
     define heap[10], no, i, j, c, root, temp;
     Read input of no of elements;
     for (i = 0; i < no; i++) /* Read input of the no's */
          scanf("%d", &heap[i]);
     for (i = 1; i < no; i++)
           c = i;
     do
     root = (c - 1) / 2;
     if (heap[root] < heap[c]) /* to create MAX heap array*/
           temp = heap[root];
          heap[root] = heap[c];
          heap[c] = temp;
          c = root;
     while (c != 0);
     /* print Heap array */
     for (i = 0; i < no; i++)
          printf("%d\t",heap[i]);
     for (j = no - 1; j >= 0; j--)
          temp = heap[0];
     heap[0] = heap[j];
/* swap max element with rightmost leaf element */
     heap[j] = temp;
     root = 0;
     do
c = 2 * root + 1; /* left node of root element */
     if ((heap[c] < heap[c + 1]) && c < j-1)
     c++;
     if (heap[root]<heap[c] && c<j)
/* again rearrange to max heap array */
          temp = heap[root];
          heap[root] = heap[c];
          heap[c] = temp;
          root = c;
          while (c < j);
          /* print the sorted array */
          for (i = 0; i < no; i++)
               printf("\t %d", heap[i]);
}
```

PROGRAM CODE:

```
#include <stdio.h>
void main(){
     int heap[10], no, i, j, c, root, temp;
     printf("\n Enter no of elements : ");
     scanf("%d", &no);
     printf("\n Enter the nos : ");
     for (i = 0; i < no; i++)
          scanf("%d", &heap[i]);
     for (i = 1; i < no; i++)
     {    c = i;
     do {
          root = (c - 1) / 2;
     if (heap[root] < heap[c])
     {    temp = heap[root];
          heap[root] = heap[c];
          heap[c] = temp;
     } c = root;
     } while (c != 0); }
     printf("\n Heap array : ");
     for (i = 0; i < no; i++)
          printf("%d\t",heap[i]);
     for (j = no - 1; j >= 0; j--)
```

```
        {   temp = heap[0];
            heap[0] = heap[j];
            heap[j] = temp;
            root = 0;
        do {
            c = 2 * root + 1;
        if ((heap[c] < heap[c + 1]) && c < j-1)
            c++;
        if (heap[root]<heap[c] && c<j)
        {   temp = heap[root];
            heap[root] = heap[c];
            heap[c] = temp;
        }
        root = c;
        } while (c < j); }
            printf("\n The sorted array is : ");
        for (i = 0; i < no; i++)
            printf("\t %d", heap[i]);
    }
```

PROGRAM OUTPUT:



```
 Enter no of elements : 5

 Enter the nos : 35 58 12 67 36

 Heap array : 67         58      12      35      36
 The sorted array is :    12      35      36      58      67
--------------------------------
Process exited after 133.4 seconds with return value 5
Press any key to continue . . .
```

QUESTION 2:

2. Menu driven C program to implement depth first search and breadth first search graph traversal algorithms

PSEUDO CODE:

```
BFS
Set all nodes to "not visited";
     q = new Queue();
     q.enqueue(initial node);
     while ( q ? empty ) do
     {    x = q.dequeue();
          if ( x has not been visited ) {
               visited[x] = true; // Visit node x
     for ( every edge (x, y) /* we are using all edges  */ )
                    if ( y has not been visited )
                    q.enqueue(y); // Use the edge (x,y)  }
}
DFS
Set all nodes to "not visited";
     s = new Stack(); // Change to use a stack
     s.push(initial node);// Push() stores a value in a stack
     while ( s ? empty ) do
     {    x = s.pop();
// Pop() remove a value from the stack
     if ( x has not been visited )
     {    visited[x] = true; // Visit node x
       for ( every edge (x, y) /* we are using all edges  */ )
              if ( y has not been visited )
              s.push(y);// Use push()  } }
```

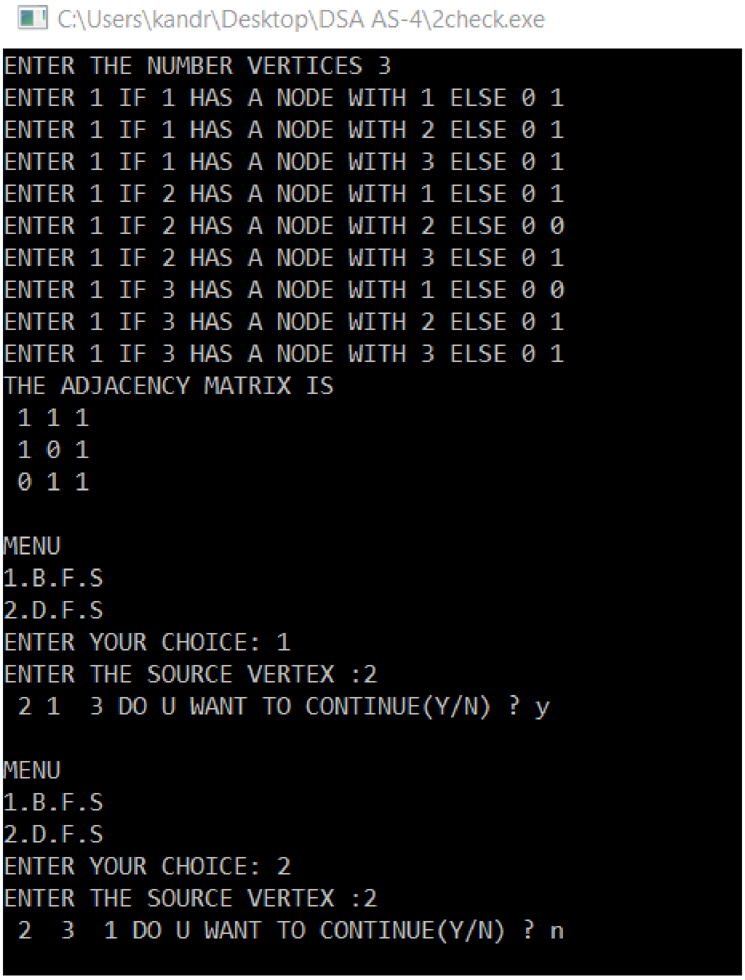
PROGRAM CODE:

```
#include<stdio.h>

int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
int delete();
void add(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();

void main()
{
int n,i,s,ch,j;
char c,dummy;
printf("ENTER THE NUMBER VERTICES ");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE 0 ",i,j);
scanf("%d",&a[i][j]);
}
}
printf("THE ADJACENCY MATRIX IS\n");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
printf(" %d",a[i][j]);
}
printf("\n");
}

do
{
for(i=1;i<=n;i++)
vis[i]=0;
printf("\nMENU");
printf("\n1.B.F.S");
printf("\n2.D.F.S");
printf("\nENTER YOUR CHOICE");
scanf("%d",&ch);
printf("ENTER THE SOURCE VERTEX :");
scanf("%d",&s);
switch(ch)
{
case 1:bfs(s,n);
break;
case 2:
dfs(s,n);
break;
}
printf("DO U WANT TO CONTINUE(Y/N) ? ");
scanf("%c",&dummy);
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
}

void bfs(int s,int n)
{
int p,i;
add(s);
vis[s]=1;
p=delete();
if(p!=0)
printf(" %d",p);
while(p!=0)
{
for(i=1;i<=n;i++)
if((a[p][i]!=0)&&(vis[i]==0))
{
add(i);
vis[i]=1;
}
p=delete();
if(p!=0)
printf(" %d ",p);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
bfs(i,n);
}
```

```
void add(int item)
{
if(rear==19)
printf("QUEUE FULL");
else
{
if(rear==-1)
{
q[++rear]=item;
front++;
}
else
q[++rear]=item;
}
}
int delete()
{
int k;
if((front>rear)||(front==-1))
return(0);
else
{
k=q[front++];
return(k);
}
}

void dfs(int s,int n)
{
int i,k;
push(s);
vis[s]=1;
k=pop();
if(k!=0)
printf(" %d ",k);
while(k!=0)
{
```

```
for(i=1;i<=n;i++)
if((a[k][i]!=0)&&(vis[i]==0))
{
push(i);
vis[i]=1;
}
k=pop();
if(k!=0)
printf(" %d ",k);
}
for(i=1;i<=n;i++)
if(vis[i]==0)
dfs(i,n);
}
void push(int item)
{
if(top==19)
printf("Stack overflow ");
else
stack[++top]=item;
}
int pop()
{
int k;
if(top==-1)
return(0);
else
{
k=stack[top--];
return(k);
}
}
```

PROGRAM OUTPUT:



C:\Users\kandr\Desktop\DSA AS-4\2check.exe

```
ENTER THE NUMBER VERTICES 3
ENTER 1 IF 1 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 1 HAS A NODE WITH 2 ELSE 0 1
ENTER 1 IF 1 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 2 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 2 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 2 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 3 HAS A NODE WITH 1 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 2 ELSE 0 1
ENTER 1 IF 3 HAS A NODE WITH 3 ELSE 0 1
THE ADJACENCY MATRIX IS
 1 1 1
 1 0 1
 0 1 1

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE: 1
ENTER THE SOURCE VERTEX :2
 2 1  3 DO U WANT TO CONTINUE(Y/N) ? y

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE: 2
ENTER THE SOURCE VERTEX :2
 2  3  1 DO U WANT TO CONTINUE(Y/N) ? n
```

QUESTION 3:

3. C program to implement Dijikstra's algorithm to find shortest path from source node to all other nodes.

PSEUDO CODE:

```
function Dijkstra(Graph, source):
      dist[source] := 0
      for each vertex v in Graph:
            if v ? source
                  dist[v] := infinity
// Unknown distance function from source to v
            previous[v] := undefined
// Previous node in optimal path from source
```

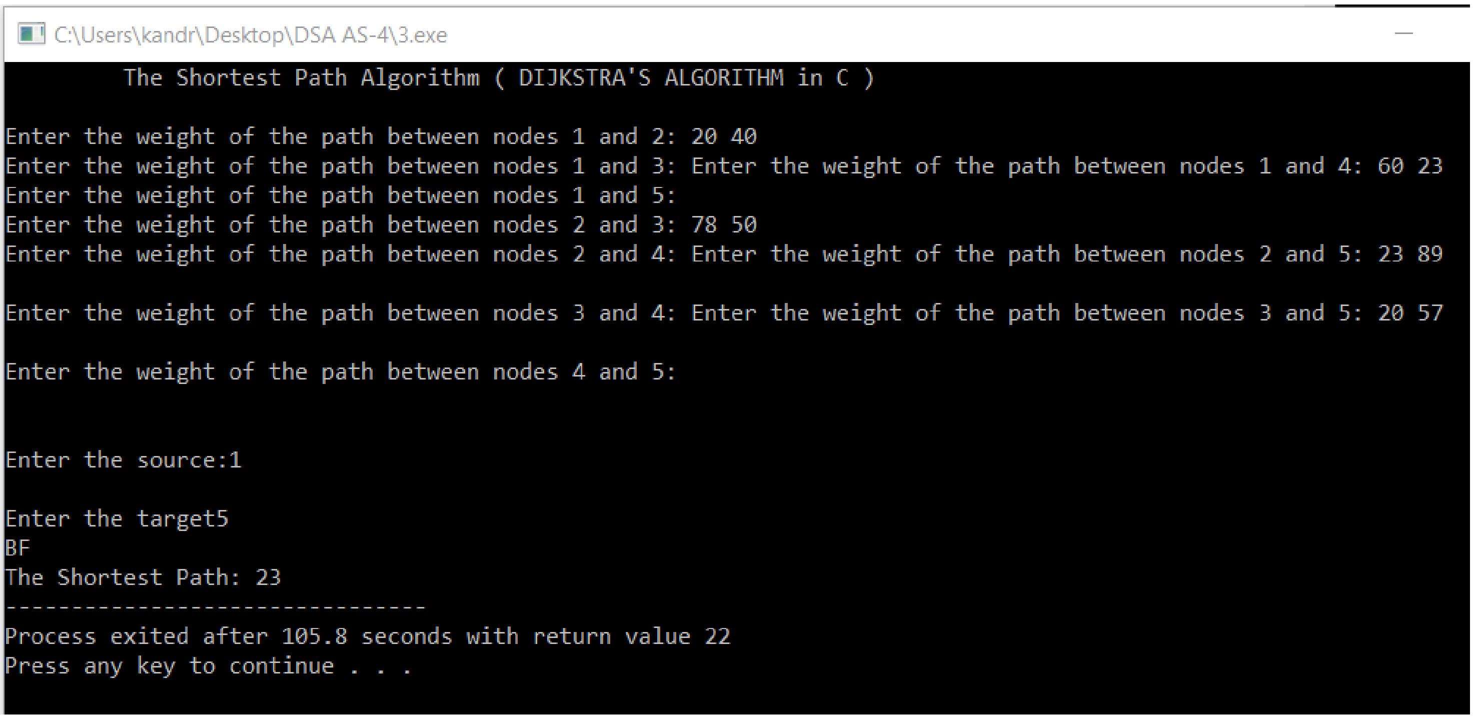
```
         end if
         add v to Q
     end for
     while Q is not empty:
         u := vertex in Q with min dist[u]
// Source node in first case
         remove u from Q
         for each neighbor v of u:
// where v has not yet been removed from Q.
         alt := dist[u] + length(u, v)
         if alt < dist[v]:
// A shorter path to v has been found
         dist[v] := alt
         previous[v] := u
         end if
     end for
end while
return dist[], previous[]
end function
```

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<string.h>
#include<math.h>
#define IN 99
#define N 6
int dijkstra(int cost[][N], int source, int target);
int dijsktra(int cost[][N],int source,int target)
{
int dist[N],prev[N],selected[N]={0},i,m,min,start,d,j;
                 char path[N];
                 for(i=1;i< N;i++)
                 {      dist[i] = IN;
                   prev[i] = -1; }
                 start = source;
                 selected[start]=1;
                 dist[start] = 0;
                 while(selected[target] ==0) {
                   min = IN; m = 0;
                   for(i=1;i< N;i++){
                        d = dist[start] +cost[start][i];
                        if(d< dist[i]&&selected[i]==0)
                   {    dist[i] = d;
                        prev[i] = start; }
                   if(min>dist[i] && selected[i]==0)
                 { min = dist[i];
                   m = i; } }
                   start = m;
                   selected[start] = 1; }
                 start = target;
                 j = 0;
                 while(start != -1)
                 { path[j++] = start+65;
                   start = prev[start]; }
                 path[j]='\0';
                 strrev(path);
                 printf("%s", path);
                 return dist[target];}
int main()
{ int cost[N][N],i,j,w,ch,co;
                 int source, target,x,y;
                 printf("\t The Shortest Path Algorithm (
DIJKSTRA'S ALGORITHM in C ) \n\n");
                 for(i=1;i< N;i++)
                 for(j=1;j< N;j++)
                 cost[i][j] = IN;
                 for(x=1;x< N;x++)
                 { for(y=x+1;y< N;y++)
                 {
                 printf("Enter the weight of the path between
nodes %d and %d: ",x,y);
                 scanf("%d",&w);
                 cost [x][y] = cost[y][x] = w; }
                 printf("\n"); }
                 printf("\nEnter the source:");
```

```
                scanf("%d", &source);
                printf("\nEnter the target");
                scanf("%d", &target);
                co = dijsktra(cost,source,target);
printf("\nThe Shortest Path: %d",co);}
```

PROGRAM OUTPUT:



```
       The Shortest Path Algorithm ( DIJKSTRA'S ALGORITHM in C )

Enter the weight of the path between nodes 1 and 2: 20 40
Enter the weight of the path between nodes 1 and 3: Enter the weight of the path between nodes 1 and 4: 60 23
Enter the weight of the path between nodes 1 and 5:
Enter the weight of the path between nodes 2 and 3: 78 50
Enter the weight of the path between nodes 2 and 4: Enter the weight of the path between nodes 2 and 5: 23 89

Enter the weight of the path between nodes 3 and 4: Enter the weight of the path between nodes 3 and 5: 20 57

Enter the weight of the path between nodes 4 and 5:


Enter the source:1

Enter the target5
BF
The Shortest Path: 23
--------------------------------
Process exited after 105.8 seconds with return value 22
Press any key to continue . . .
```

QUESTION 4:

4. Menu driven C program to implement insertion, selection and bubble sort.

PSEUDO CODE:

```
BUBBLE SORT
For I = 0 to N - 2
     For J = 0 to N - 2
          If (A(J) > A(J + 1)
          Temp = A(J)
          A(J) = A(J + 1)
          A(J + 1) = Temp
     End-If
     End-For
End-For
```

SELECTION SORT

```
For I = 0 to N-1 do:
     Smallsub = I
     For J = I + 1 to N-1 do:
          If A(J) < A(Smallsub)
          Smallsub = J
          End-If
     End-For
     Temp = A(I)
     A(I) = A(Smallsub)
     A(Smallsub) = Temp
End-For
```

INSERTION SORT

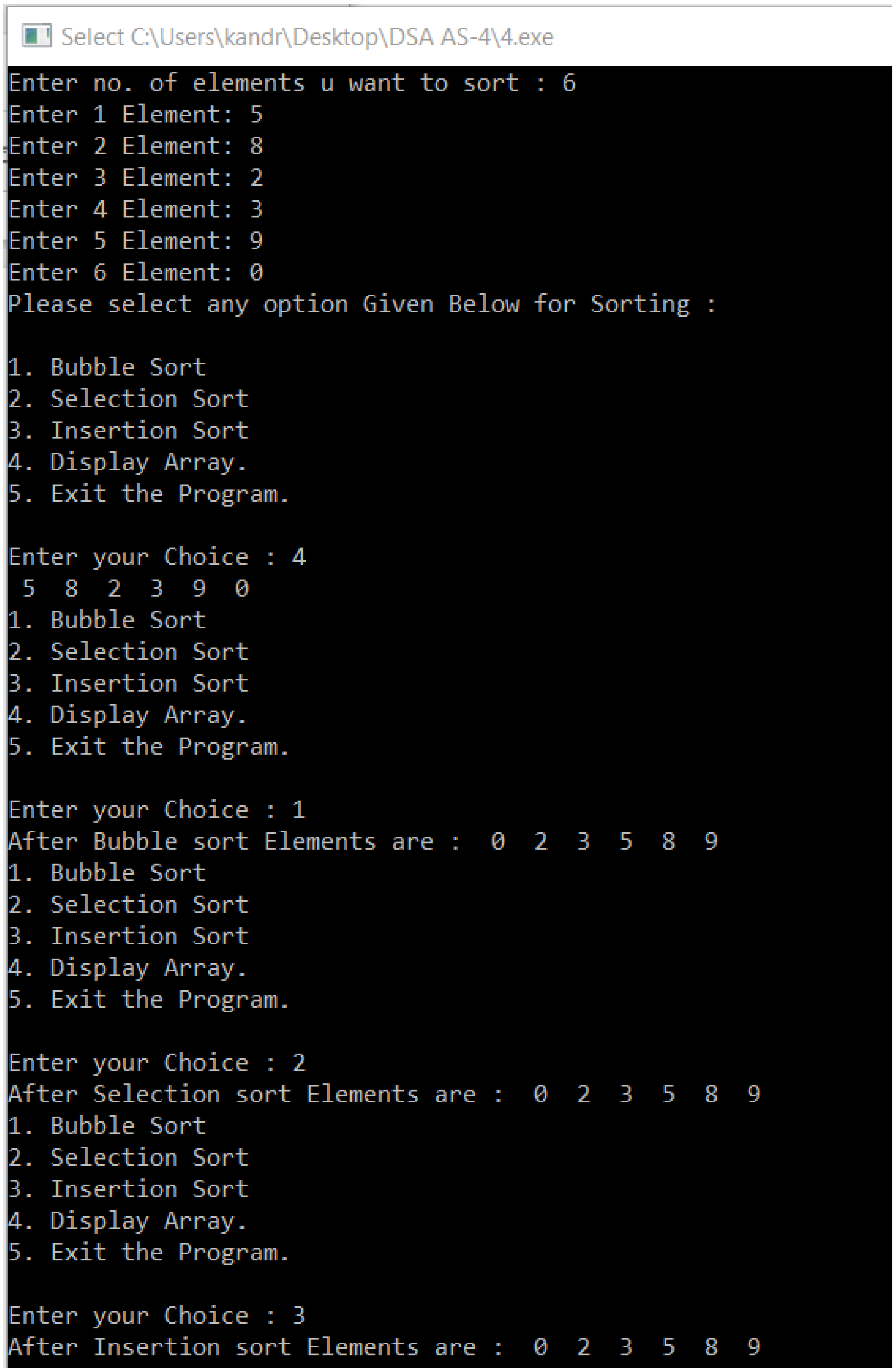
```
For I = 1 to N-1
     J = I
     Do while (J > 0) and (A(J) < A(J - 1)
          Temp = A(J)
          A(J) = A(J - 1)
          A(J - 1) = Temp
          J = J - 1
     End-Do
End-For
```

PROGRAM CODE:

```
#include<stdio.h>
#include<stdlib.h>
void display(int a[],int n);
void bubble_sort(int a[],int n);
void selection_sort(int a[],int n);
void insertion_sort(int a[],int n);

int main()
{ int n,choice,i;
char ch[20];
printf("Enter no. of elements u want to sort : ");
scanf("%d",&n);
int arr[n];
for(i=0;i<n;i++){
                 printf("Enter %d Element: ",i+1);
                 scanf("%d",&arr[i]); }
                 printf("Please select any option Given Below
for Sorting : \n");
while(1){
                 printf("\n1. Bubble Sort\n2. Selection
Sort\n3. Insertion Sort\n4. Display Array.\n5. Exit the
Program.\n");
                 printf("\nEnter your Choice : ");
                 scanf("%d",&choice);
                 switch(choice){
                 case 1:
                   bubble_sort(arr,n);
                   break;
                 case 2:
                   selection_sort(arr,n);
                   break;
                 case 3:
                   insertion_sort(arr,n);
                   break;
                 case 4:
                   display(arr,n);
                   break;
                 case 5:
                   return 0;
                 default:
                   printf("\nPlease Select only 1-5 option ----
\n"); }}
                 return 0;}

void display(int arr[],int n)
                 { int i;
for(i=0;i<n;i++)
                 { printf(" %d ",arr[i]); }}

void bubble_sort(int arr[],int n){
                 int i,j,temp;
for(i=0;i<n;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;
} } }
                 printf("After Bubble sort Elements are : ");
                 display(arr,n);
}
void selection_sort(int arr[],int n)
{ int i,j,temp;
for(i=0;i<n-1;i++){
for(j=i+1;j<n;j++){
if(arr[i]>arr[j])
                 { temp=arr[i];
                 arr[i]=arr[j];
                 arr[j]=temp; } } }
                 printf("After Selection sort Elements are :
");
                 display(arr,n);}

void insertion_sort(int arr[],int n)
                 { int i,j,min;
for(i=1;i<n;i++)
                 { min=arr[i];
                 j=i-1;
while(min<arr[j] && j>=0)
                 { arr[j+1]=arr[j];
                 j=j-1; }
                 arr[j+1]=min; }
                 printf("After Insertion sort Elements are :
");
                 display(arr,n);
}
```

PROGRAM OUTPUT:



```
Enter no. of elements u want to sort : 6
Enter 1 Element: 5
Enter 2 Element: 8
Enter 3 Element: 2
Enter 4 Element: 3
Enter 5 Element: 9
Enter 6 Element: 0
Please select any option Given Below for Sorting :

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Display Array.
5. Exit the Program.

Enter your Choice : 4
 5  8  2  3  9  0
1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Display Array.
5. Exit the Program.

Enter your Choice : 1
After Bubble sort Elements are :  0  2  3  5  8  9
1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Display Array.
5. Exit the Program.

Enter your Choice : 2
After Selection sort Elements are :  0  2  3  5  8  9
1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Display Array.
5. Exit the Program.

Enter your Choice : 3
After Insertion sort Elements are :  0  2  3  5  8  9
```

QUESTION 5:

5. Menu driven C program to implement quick and merge sort.

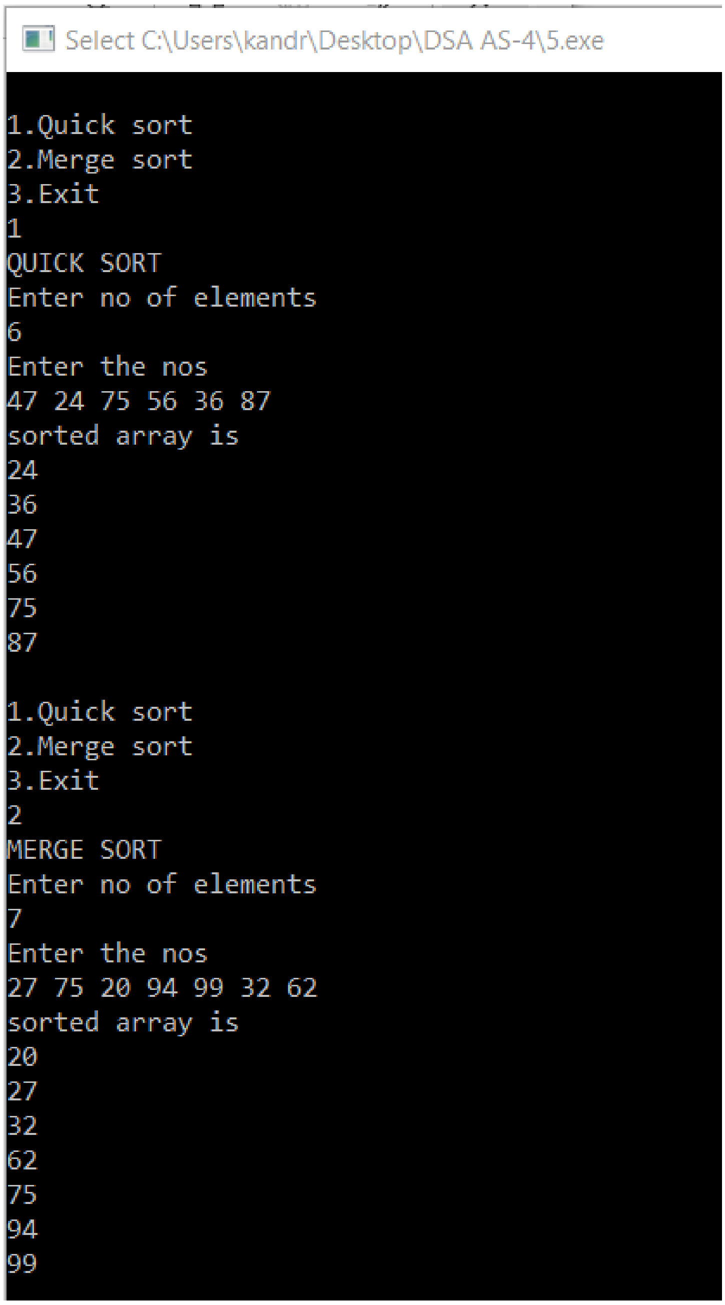
PSEUDO CODE:

```
pivot = a[center];
i = from;
j = to-1;
while (true) {
     while (a[++i] < pivot) { }
// move i up and keep going if element is < pivot
     while (pivot < a[--j]) { }
// move j down and keep going if element is > pivot
     if (i < j) { // swap a[i] and a[j]
          int tmp = a[i];
          a[i] = a[j];
          a[j] = tmp; }
     else { break; }}
CALLING RECURSIVELY QUICKSORT
quickSort(a, from, i); // include from, exclude i
quickSort(a, i, to); // include i, exclude to
CALLING MERGESORT RECURSIVELY
mergeSort(a) {
     if (a.length == 1) {
          return; }
split a into two equal halves: a = (a1,a2)
mergeSort(a1)
mergeSort(a2)
// then, the merge step, where we actually do something
// by merging the two sorted halves back into the full array:
A = merge(a1,a2)}
```

PROGRAM CODE:

```
#include<stdio.h>
#include<conio.h>
void quick_sort(int[],int,int);
int partition(int[],int,int);
void mergesort(int[],int,int);
void merge(int[],int,int,int,int);
void quick_sort(int a[100],int l,int u)
{int j;
if(l<u)
{j=partition(a,l,u);
quick_sort(a,l,j-1);
quick_sort(a,j+1,u);}}
int partition(int a[100],int l,int u)
{int v,i,j,temp;
v=a[l];
i=l;
j=u+1;
do{
do{
i++;}
while(a[i]<v&&i<=u);
do{
j--;}
while(a[j]>v);
if(i<j)
{temp=a[i];
a[i]=a[j];
a[j]=temp;}}
while(i<j);
a[l]=a[j];
a[j]=v;
return(j);}
void mergesort(int a[100],int i, int j)
{int mid;
if(i<j)
{mid=(i+j)/2;
mergesort(a,i,mid);
mergesort(a,mid+1,j);
merge(a,i,mid,mid+1,j);}}
void merge(int a[100],int i1,int j1, int i2,int j2)
{int temp[100];
int i,j,k;
i=i1;j=i2;k=0;
while(i<=j1&&j<=j2)
{if(a[i]<a[j])
temp[k++]=a[i++];
else
temp[k++]=a[j++];}
while(i<=j1)
temp[k++]=a[i++];
while(j<=j2)
temp[k++]=a[j++];
for(i=i1,j=0;i<=j2;i++,j++)
a[i]=temp[j];}
void main()
{int a[100],n,i,op;
while(1)
{printf("\n1.Quick sort\n2.Merge sort\n3.Exit\n");
scanf("%d",&op);
switch(op)
{case 1:printf("QUICK SORT\n");
                 printf("Enter no of elements\n");
                 scanf("%d",&n);
                 printf("Enter the nos\n");
                 for(i=0;i<=n-1;i++)
                 scanf("%d",&a[i]);
                 quick_sort(a,0,n-1);
                 printf("sorted array is\n");
                 for(i=0;i<=n-1;i++)
                 printf("%d\n",a[i]);
                 break;
case 2:printf("MERGE SORT\n");
                 printf("Enter no of elements\n");
                 scanf("%d",&n);
                 printf("Enter the nos\n");
                 for(i=0;i<=n-1;i++)
                 scanf("%d",&a[i]);
                 mergesort(a,0,n-1);
                 printf("sorted array is\n");
                 for(i=0;i<=n-1;i++)
                 printf("%d\n",a[i]);
                 break;
case 3:exit(1);
default:printf("Invalid Choice\n");}}}
```

PROGRAM OUTPUT:



```
1.Quick sort
2.Merge sort
3.Exit
1
QUICK SORT
Enter no of elements
6
Enter the nos
47 24 75 56 36 87
sorted array is
24
36
47
56
75
87

1.Quick sort
2.Merge sort
3.Exit
2
MERGE SORT
Enter no of elements
7
Enter the nos
27 75 20 94 99 32 62
sorted array is
20
27
32
62
75
94
99
```

- -