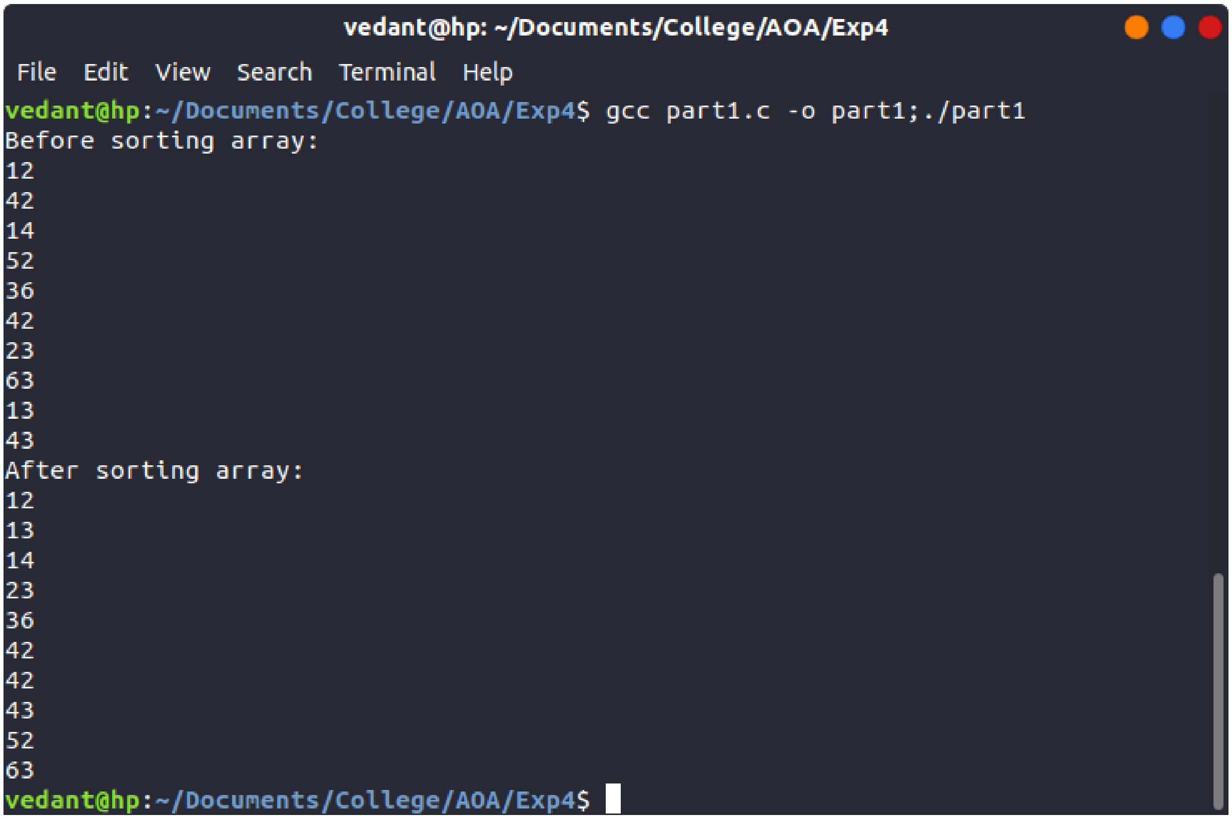
Name:Vedant Mhatre

Moodle ID: 18102055

Roll Number: 31

Modification 1:

```
#include<stdio.h>
void mergesort(int[],int,int);
void mergearray(int[],int,int,int);

void sort(int arr[],int first, int last, int arr2[])
{
        int middle=first+last/2;
        int i=first,j=middle,k=0;

        while(i<middle && j<last)
        {
                if(arr[i] < arr[j])
                        arr2[k++] = arr[i++];
                else
                        arr2[k++] = arr[j++];
        }

        while(i<middle)
                arr2[k++] = arr[i++];

        while(j<last)
                arr2[k++] = arr[j++];

}


void quicksort(int x[],int first,int last)
{
        int pivot,i,j,t;
        if(first<last)
        {
                pivot=first;
                i=first;
                j=last;
                while(i<j)
                {
                        while(x[i]<=x[pivot]&&i<last)
                i++;
                while(x[j]>x[pivot])
                j--;
                if(i<j)
                {
                        t=x[i];
                        x[i]=x[j];
                        x[j]=t;
                }
            }
            t=x[pivot];
            x[pivot]=x[j];
            x[j]=t;
            quicksort(x,first,j-1);
            quicksort(x,j+1,last);
        }
}

void mergesort(int a[],int beg,int end)
{
int mid;
if(beg<end)
{
        mid=(beg+end)/2;
        mergesort(a,beg,mid);
        mergesort(a,mid+1,end);
        mergearray(a,beg,mid,end);
}
}
void mergearray(int a[],int beg,int mid,int end){
        int i,leftend,num,temp,j,k,b[50];
        for(i=beg;i<=end;i++)
            b[i]=a[i];
        i=beg;
        j=mid+1;
        k=beg;
        while((i<=mid)&&(j<=end)) {
            if(b[i]<=b[j])
            {
                a[k]=b[i];
                i++;
                k++;
            }
            else  {
                a[k]=b[j];
                j++;
                k++;
            }
        }

        if(i<=mid){
              while(i<=mid) {
                a[k]=b[i];
                i++;
                k++;
            }
        }
        else {
            while(j<=end){
                a[k]=b[j];
                j++;
                k++;
            }
        }
}

void main()
{
        int arr[10]={12,42,14,52,36,42,23,63,13,43};
        int arr2[10];
        printf("Before sorting array:\n");
        for(int i=0;i<10;i++)
            printf("%d\n",arr[i]);

        quicksort(arr,0,4);
        mergesort(arr,5,9);
        sort(arr,0,10,arr2);

        printf("After sorting array:\n");
        for(int i=0;i<10;i++)
            printf("%d\n",arr2[i]);
}
```


vedant@hp: ~/Documents/College/AOA/Exp4
File Edit View Search Terminal Help

```
vedant@hp:~/Documents/College/AOA/Exp4$ gcc part1.c -o part1;./part1
Before sorting array:
12
42
14
52
36
42
23
63
13
43
After sorting array:
12
13
14
23
36
42
42
43
52
63
vedant@hp:~/Documents/College/AOA/Exp4$ 
```

Modification 2:

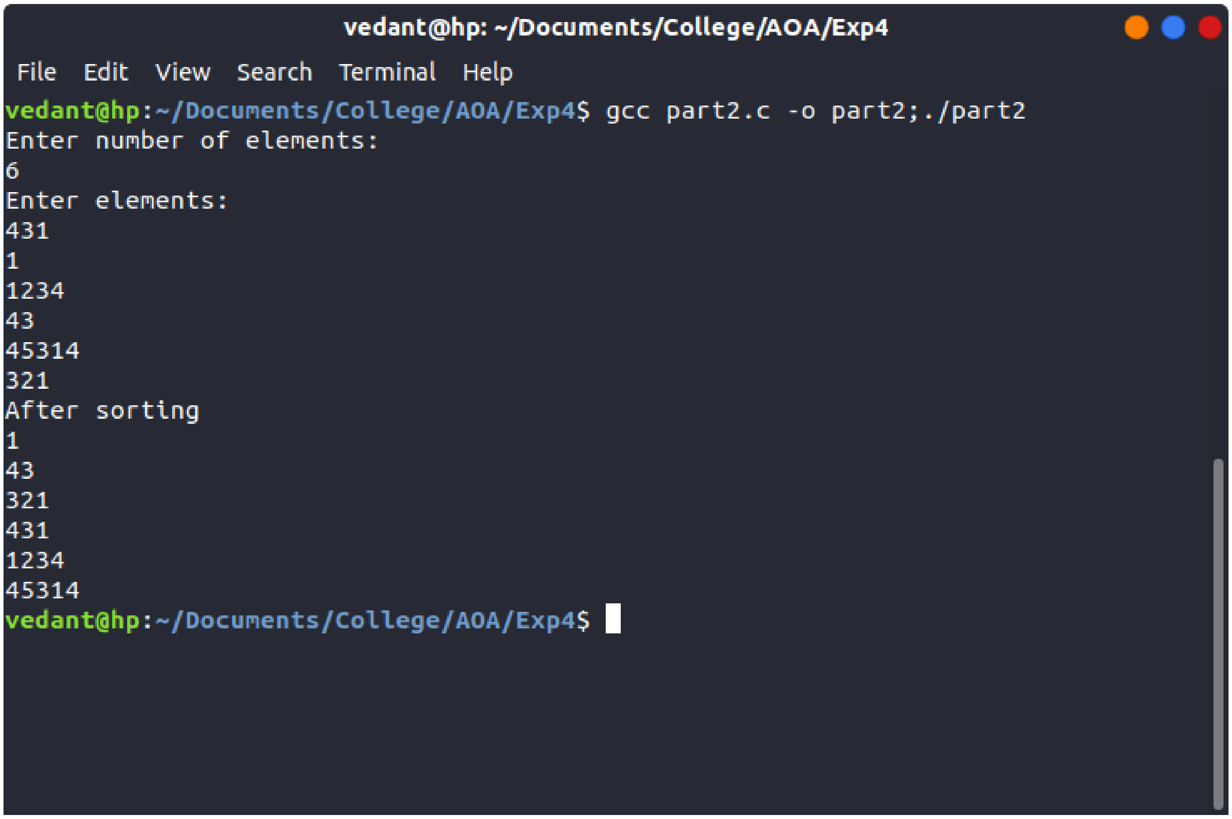
```
#include<stdio.h>
#include<stdlib.h>

struct node{
	int data;
	int id;
	struct node *next;
};

struct node *top = NULL;
struct node *ptr = NULL;

void insert()
{
	struct node *new_node = (struct node *) malloc(sizeof(struct node));
```

```
        scanf("%d",&new_node->data);
        new_node->next = NULL;

        if (top == NULL){
                new_node->id = 0;
                top = new_node;
                return;
        }
        else{
                struct node *last = top;
                while(last->next != NULL)
                        last = last->next;
                new_node->id = (last->id)+1;
                last->next = new_node;
                return;
        }
}

void print()
{
        ptr = top;
        while(ptr!=NULL)
        {
                printf("%d\n",ptr->data);
                ptr = ptr->next;
        }
}

void ptrAtId(int pid)
{
        ptr = top;
        while(ptr->id!=pid)
                ptr = ptr->next;
```

```
}

int ptrData(int pid)
{
        ptrAtId(pid);
        return ptr->data;
}

void changeData(int pid,int value)
{
        ptrAtId(pid);
        ptr->data = value;
}

void mergearray(struct node *top,int beg,int mid,int end){
        int i,j,k,b[50];

        for(i=beg;i<=end;i++)
                b[i]=ptrData(i);

        i=beg;
        j=mid+1;
        k=beg;

        while((i<=mid)&&(j<=end)) {
                if(b[i]<=b[j])
                {
                        changeData(k,b[i]);
                        i++;
                        k++;
                }
                else  {
                        changeData(k,b[j]);
                        j++;
                        k++;
                    }
            }

            if(i<=mid){
                     while(i<=mid) {
                        changeData(k,b[i]);
                        i++;
                        k++;
                    }
            }
            else {
                    while(j<=end){
                        changeData(k,b[j]);
                        j++;
                        k++;
                    }
            }

    }

    void mergesort(struct node *top,int beg,int end)
    {
    int mid;
    if(beg<end)
    {
            mid=(beg+end)/2;
            mergesort(top,beg,mid);
            mergesort(top,mid+1,end);
            mergearray(top,beg,mid,end);
    }
    }
```

```
void sort(struct node *top,int first, int last, int arr2[])
{
    int middle=first+last/2;
    int i=first,j=middle,k=0;

    while(i<middle && j<last)
    {
        if(ptrData(i) < ptrData(j))
            arr2[k++] = ptrData(i++);
        else
            arr2[k++] = ptrData(j++);
    }

    while(i<middle)
        arr2[k++] = ptrData(i++);

    while(j<last)
        arr2[k++] = ptrData(j++);

}


void quicksort(struct node *top,int first,int last)
{
    int pivot,i,j,t;
    if(first<last)
    {
        pivot=first;
        i=first;
        j=last;
        while(i<j)
        {
            while(ptrData(i)<=ptrData(pivot)&&i<last)
            i++;
            while(ptrData(j)>ptrData(pivot))
            j--;
            if(i<j)
            {
                t=ptrData(i);
                changeData(i,ptrData(j));
                changeData(j,t);
            }
        }

        t=ptrData(pivot);
        changeData(pivot,ptrData(j));
        changeData(j,t);

        quicksort(top,first,j-1);
        quicksort(top,j+1,last);

    }
}



void main()
{
    int n;
    printf("Enter number of elements:\n");
    scanf("%d",&n);

    printf("Enter elements:\n");
```

```
     for (int i = 0; i < n; i++)
    insert();

     quicksort(top,0,(n/2)-1);
     mergesort(top,(n/2),n-1);
     int arr2[n];
     sort(top,0,n,arr2);
     printf("After sorting\n");
     for(int i=0;i<n;i++)
             printf("%d\n",arr2[i]);

}
```



```
vedant@hp:~/Documents/College/AOA/Exp4$ gcc part2.c -o part2;./part2
Enter number of elements:
6
Enter elements:
431
1
1234
43
45314
321
After sorting
1
43
321
431
1234
45314
vedant@hp:~/Documents/College/AOA/Exp4$
```