DAA LAB ASSIGNMENT-5

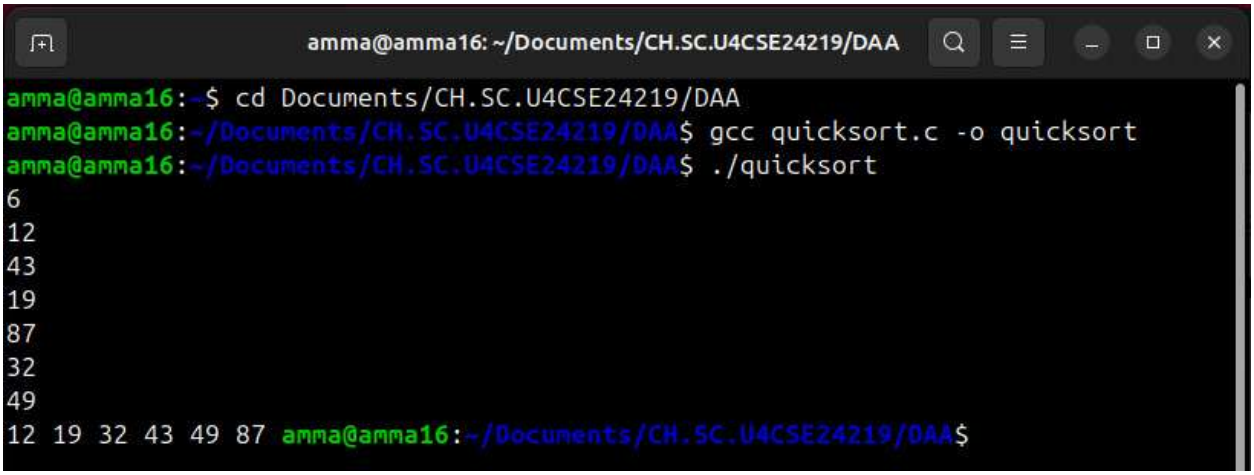
1.QUICK SORT:

CODE:

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

void swap(int *a, int *b) { int t = *a;

 *a = *b;

 *b = t; }

int partition(int a[], int low, int high) {

 int pivot = a[high];

 int i = low - 1;

for (int j = low; j < high; j++)
{
   if (a[j] <= pivot)
   {
      i++;
      swap(&a[i], &a[j]);
   }
}
swap(&a[i + 1], &a[high]);
return i + 1;

}

void quickSort(int a[], int low, int high) {

if (low < high) {

 int pi = partition(a, low, high);

 quickSort(a, low, pi - 1);
```

```
 quickSort(a, pi + 1, high);

}

 }

int main() {

int n;

scanf("%d", &n);

int *a = (int *)malloc(n * sizeof(int));
for (int i = 0; i < n; i++)
   scanf("%d", &a[i]);
quickSort(a, 0, n - 1);
for (int i = 0; i < n; i++)
   printf("%d ", a[i]);
free(a);
return 0;

}
```

OUTPUT:



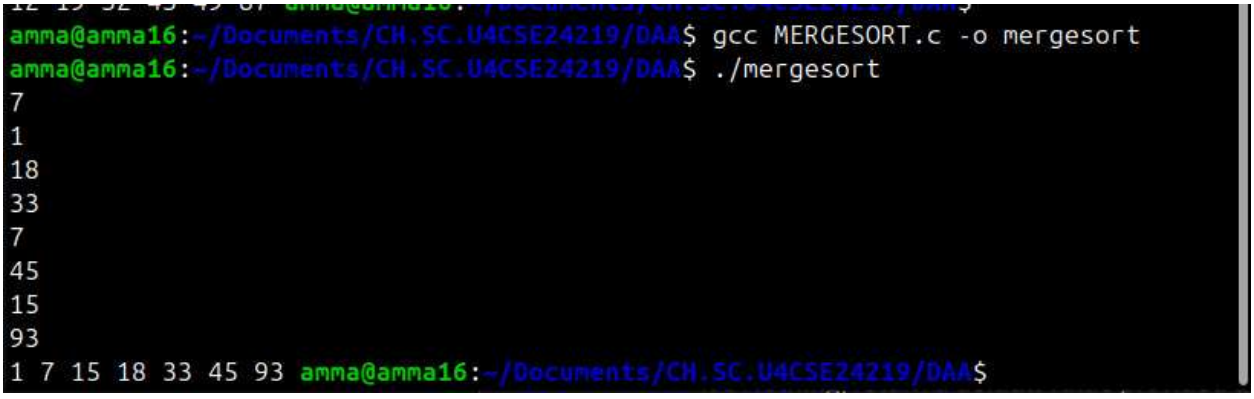
2.MERGE SORT

CODE:

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

void merge(int a[], int l, int m, int r) {

int i = l, j = m + 1, k = 0;

 int *temp = (int *)malloc((r - l + 1) * sizeof(int));

while (i <= m && j <= r)
{
   if (a[i] <= a[j])
      temp[k++] = a[i++];
   else
      temp[k++] = a[j++];
}
while (i <= m)
   temp[k++] = a[i++];
while (j <= r)
   temp[k++] = a[j++];
for (i = l, k = 0; i <= r; i++, k++)
   a[i] = temp[k];
free(temp);

}

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

 if (l < r) {

 int m = l + (r - l) / 2;

mergeSort(a, l, m);

mergeSort(a, m + 1, r);

 merge(a, l, m, r);
```

```
 } }

int main() {

int n; scanf("%d", &n);

 int a[n];

for (int i = 0; i < n; i++)
    scanf("%d", &a[i]);
mergeSort(a, 0, n - 1);
for (int i = 0; i < n; i++)
    printf("%d ", a[i]);
return 0;}
```

OUTPUT:



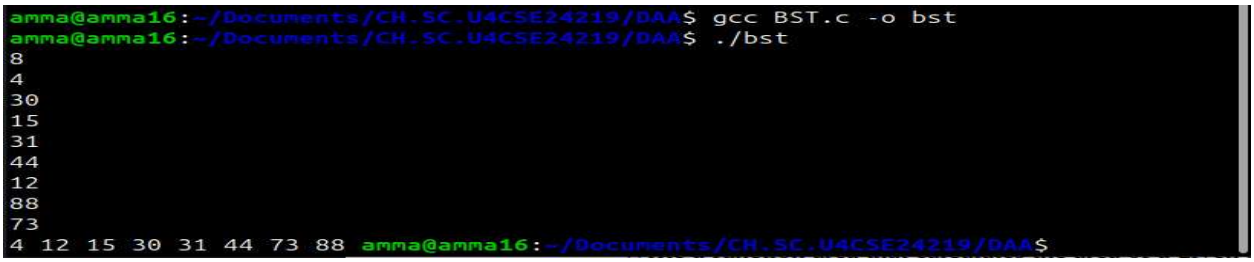
3)BST

CODE:

```
#include <stdio.h>
 #include <stdlib.h>
struct Node {
 int data;
struct Node *left, *right;
 };
struct Node* newNode(int data) {
struct Node* node = (struct Node*)malloc(sizeof(struct Node));
 node->data = data;
 node->left = node->right = NULL;
 return node; }
struct Node* insert(struct Node* root, int data)
{                                   if (root == NULL) return newNode(data);
if (data < root->data)

else if (data > root->data)
   root->right = insert(root->right, data);
return root;}
void inorder(struct Node* root) {
 if (root != NULL) {
inorder(root->left);
printf("%d ", root->data);
 inorder(root->right); } }
int main() {
```

```
 struct Node* root = NULL;

 int n, x;

scanf("%d", &n);
for (int i = 0; i < n; i++)
{
   scanf("%d", &x);
   root = insert(root, x);
}
inorder(root);
return 0;

}
```

OUTPUT:



KAMIREDDYGARI NAGA GNANESWARA REDDY

CH.SC.U4CSE24219