DSA DIGITAL ASSESSMENT – 5

Done By: Sparsh Arya

Registration number: 17bec0656

1. Menu driven C program to create binary search tree. Perform insertion and deletion operations. Display the contents of BST using preorder, inorder and postorder traversal.

PSEUDO CODE*:*

```
struct node{
  int data;
  struct node *left, *right;
};


newNode(int key){
  struct node* temp = malloc(sizeof(struct node)) temp-
  >data = key
  temp->left = NULL
  temp->right = NULL
  return temp
}


insert(struct node* root, int data){ if root
  == NULL
          return newNode(data) if
  data > root->data
```

```
    root->right = insert(root->right, data) else if
  data < root->data
    root->left = insert(root->left, data) return
  root
}

**maxValueNode**(struct node* root){
  current = root
  while(current)
    current = current->right
  return current
}

**deleteNode**(struct node* root, int key){ if root
  == NULL
          return root
  if key < root->data
    root->left = deleteNode(root->left, key) else
  if key > root->data
    root->right = deleteNode(root->right, key) else{
    if(!root->left){
      temp = root->right
      free(root)
      return temp
    }

    else if(!root->right){

      temp = root->left

      free(root)

      return temp

    }


    temp = maxValueNode(root->left) root-

    >data = temp->data

    root->left = deleteNode(root->left, temp->data)

  }


  return root

}
```

postOrder(struct node* root){

```
  if(!root) return postOrder(root-

  >left) postOrder(root->right)

  print root->data

}
```

preOrder(struct node* root){

```
  if(!root) return

  print  root->data)
  preOrder(root->left)

  preOrder(root->right)

}


inOrder(struct node* root){

  if(!root) return postOrder(root-

  >left) print root->data

  postOrder(root->right)

}


main(){

  root  =  NULL

  while(1){

    print("1.        Insert\n");

    print("2.       Delete\n");

    print("3. Print\n");

    print("4. Exit\n");

    input choice

    switch(choice){

      case 1: print("Enter a number: \n") input n)

              root = insert(root, n)

              break

      case 2: print("\nEnter the number you wish to delete: \n")
```

```
                  input n
              root = deleteNode(root, n)
              print(" deleted!\n") break;
      case 3: print("1. Inorder\n")
              print("2. Postorder\n")
              print("3. Preorder\n") input
              printChoice if(printChoice
              == 1){
                inOrder(root)

              } else if(printChoice == 2){
                postOrder(root)
              }
              else if(printChoice == 3){
                preOrder(root)
              }
              else print("Invalid input!")
              break
      case 4: exit(0)
      default:print("Invalid input!")

   }

  }
  return 0;
```

```
}
```

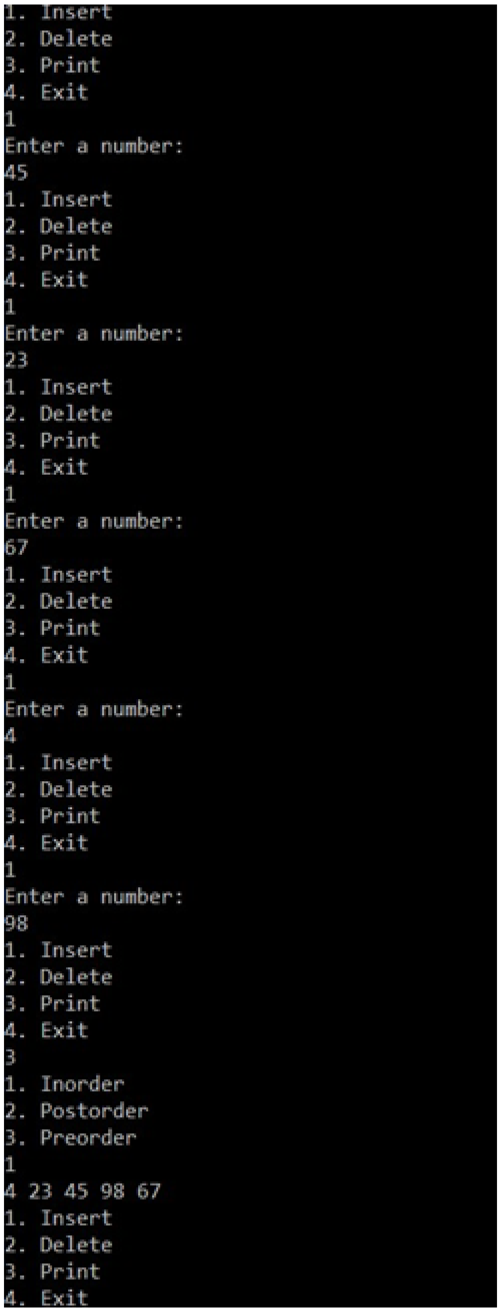
Code:

```
#include<stdlib.h>

#include<stdio.h>


struct node{ int

    data;

    struct node *left, *right;

};


struct node* newNode(int key){

    struct node* temp = (struct node*)malloc(sizeof(structnode)); temp->data = key;

    temp->left = NULL; temp-

    >right = NULL; return

    temp;

}


struct node* insert(struct node* root, int data){ if(!root) return

    newNode(data);

    if(data > root->data)

        root->right = insert(root->right, data); else if(data < root-

    >data)

        root->left = insert(root->left, data); return root;

}
```

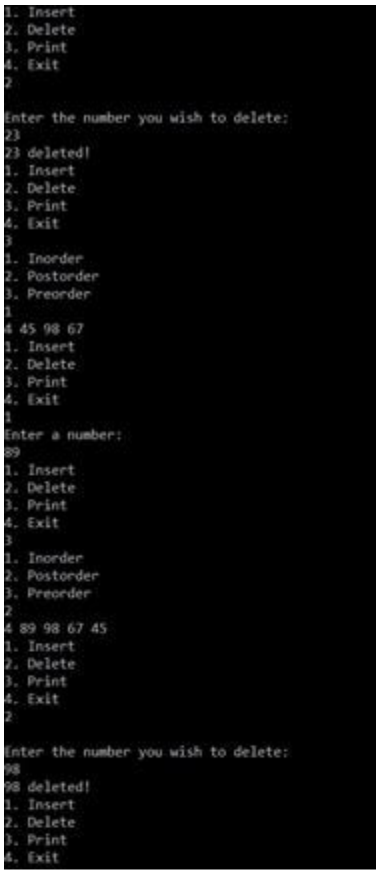
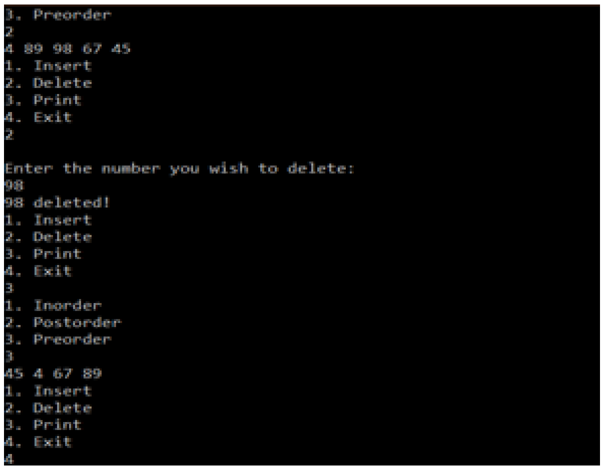
```
struct node* maxValueNode(struct node* root){ struct node*
    current = root;
    while(current)
        current = current->right; return
    current;
}

struct node* deleteNode(struct node* root, int key){ if(!root) return root;
    if(key < root->data)
        root->left = deleteNode(root->left, key); else if(key > root-
    >data)
        root->right = deleteNode(root->right, key); else{
        if(!root->left){
            struct node* temp = root->right; free(root);
            return temp;
        }
        else if(!root->right){
            struct node* temp = root->left; free(root);
            return temp;
        }

        struct node* temp = maxValueNode(root->left);
        root->data = temp->data;

        root->left = deleteNode(root->left, temp->data);



    }


    return root;

}


void postOrder(struct node* root){ if(!root) return;

    postOrder(root->left); postOrder(root-

    >right); printf("%d ", root->data);

}


void preOrder(struct node* root){ if(!root)

    return;

    printf("%d ", root->data);

    preOrder(root->left); preOrder(root-

    >right);

}


void inOrder(struct node* root){ if(!root)

    return; postOrder(root->left); printf("%d ",

    root->data);
    postOrder(root->right);
}


int main(){
    int choice, n, printChoice; struct node*
    root = NULL; while(1){
        printf("1. Insert\n"); printf("2.
        Delete\n"); printf("3. Print\n");
        printf("4. Exit\n"); scanf("%d",
        &choice); switch(choice){
            case 1: printf("Enter a number: \n"); scanf("%d", &n);
                            root = insert(root, n); break;
            case 2: printf("\nEnter the number you wish to delete:
\n");
                            scanf("%d", &n);
                            root  =  deleteNode(root,  n);  printf("%d
                            deleted!\n", n); break;
            case 3: printf("1. Inorder\n"); printf("2.
                            Postorder\n"); printf("3. Preorder\n");
                            scanf("%d", &printChoice);
                            if(printChoice == 1){
```

```
                                        inOrder(root);

                                        printf("\n");

                                    }

                                    else if(printChoice == 2){ postOrder(root);

                                        printf("\n");

                                    }

                                    else if(printChoice == 3){ preOrder(root);

                                        printf("\n");

                                    }

                                    else printf("Invalid input!"); break;

                    case 4: exit(0); default:printf("Invalid input!");


                }


            }

            return 0;


        }
```

OUTPUT:



```
1. Insert
2. Delete
3. Print
4. Exit
1
Enter a number:
45
1. Insert
2. Delete
3. Print
4. Exit
1
Enter a number:
23
1. Insert
2. Delete
3. Print
4. Exit
1
Enter a number:
67
1. Insert
2. Delete
3. Print
4. Exit
1
Enter a number:
4
1. Insert
2. Delete
3. Print
4. Exit
1
Enter a number:
98
1. Insert
2. Delete
3. Print
4. Exit
3
1. Inorder
2. Postorder
3. Preorder
1
4 23 45 98 67
1. Insert
2. Delete
3. Print
4. Exit
```


```
1. Insert
2. Delete
3. Print
4. Exit
2

Enter the number you wish to delete:
23
23 deleted!
1. Insert
2. Delete
3. Print
4. Exit
3
1. Inorder
2. Postorder
3. Preorder
1
4 45 98 67
1. Insert
2. Delete
3. Print
4. Exit
1
Enter a number:
89
1. Insert
2. Delete
3. Print
4. Exit
3
1. Inorder
2. Postorder
3. Preorder
2
4 89 98 67 45
1. Insert
2. Delete
3. Print
4. Exit
2

Enter the number you wish to delete:
98
98 deleted!
1. Insert
2. Delete
3. Print
4. Exit
```

```
3. Preorder
2
4 89 98 67 45
1. Insert
2. Delete
3. Print
4. Exit
2

Enter the number you wish to delete:
98
98 deleted!
1. Insert
2. Delete
3. Print
4. Exit
3
1. Inorder
2. Postorder
3. Preorder
3
45 4 67 89
1. Insert
2. Delete
3. Print
4. Exit
4
```

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

PSEUDO CODE:

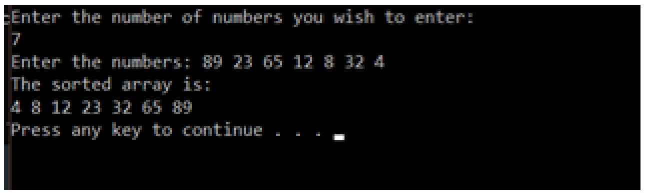
```
swap(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;}

heapify(int arr[], int n, int i){
    largest = i
    l = 2 * i + 1 r
    = 2 * i + 2
    if(l < n && arr[l] > arr[largest])
        largest = l
    if(r < n && arr[r] > arr[largest])
        largest = r
    if(largest != i){
        swap(&arr[i], &arr[largest])
        heapify(arr, n, largest)
    }
heapSort(int arr[], int n){ for
    i = n/2 – 1to 0:
        heapify(arr, n, i)
    for i=n-1 to 0:{
        swap(&arr[0], &arr[i])
        heapify(arr, i, 0)
    }
}



int main(){
    int n;
    print("Enter the number of numbers you wish to enter: \n") input n
    int arr[50]
    print("Enter the numbers: ")
    for(int i = 0; i < n; i++)
        input arr[i])
    heapSort(arr, n)
    print("The sorted array is: \n") for i =
    0 to n:
        print arr[i])
    return 0
```

Code:

```
#include<stdio.h>

#include<stdlib.h>


void swap(int *a, int *b){ int temp;

        temp = *a;

        *a = *b;

        *b = temp;

}


void heapify(int arr[], int n, int i){ int largest = i;

        int l = 2 * i + 1; int r = 2 *

        i + 2;

        if(l < n && arr[l] > arr[largest]) largest = l;

        if(r < n && arr[r] > arr[largest]) largest = r;

        if(largest != i){ swap(&arr[i], &arr[l]);

                heapify(arr, n, largest);

        }
```

```
void heapSort(int arr[], int n){ for(int i = n/2 - 1; i >=
    0; i--)
        heapify(arr, n, i); for(int i=n-1; i >
    0; i--){
        swap(&arr[0], &arr[i]);
        heapify(arr, i, 0);
    }
}


int main(){
    int n;
    printf("Enter the number of numbers you wish to enter: \n"); scanf("%d", &n);
    int arr[50];
    printf("Enter the numbers: "); for(int i = 0; i
    < n; i++)
        scanf("%d", &arr[i]);
    heapSort(arr, n);
    printf("The sorted array is: \n"); for(int i = 0; i < n;
    i++)
        printf("%d ", arr[i]); return 0;
```

OUTPUT:



```
Enter the number of numbers you wish to enter:
7
Enter the numbers: 89 23 65 12 8 32 4
The sorted array is:
4 8 12 23 32 65 89
Press any key to continue . . .
```