L11+L12

Data Structures and Algorithms: CSE2003

**Name:** Allen Ben Philipose

**Reg No:** 18BIS0043

Activity – 3

**Question**

Devise code in C to accept age of n employees and construct a min heap. Then display the employee’s ages from highest to smallest. While constructing min heap display each insertion after heapify

**Code**

#include<stdio.h>

#include<string.h>

void heapify(int a[], int n, int i)

{

int s = i;

int left = 2\*i+1;

int right = 2\*i+2;

if(left<n && a[left]<a[s])

s=left;

if(right<n && a[right]<a[s])

s=right;

if(s!=i)

{

int z = a[i];

a[i] = a[s];

a[s] = z;

heapify(a,n,s);

}

}

void sort(int a[], int n)

{

int i;

for(i = n/2-1; i>=0; i--)

heapify(a,n,i);

printf("Heap: ");

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

printf("\n%d",a[i]);

for(i=n-1;i>=0;i--)

{

int z = a[0];

a[0] = a[i];

a[i] = z;

heapify(a,i,0);

printf("\nInserting...");

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

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

}

}

void main()

{

int i,x;

printf("Enter the number of employees: ");

scanf("%d",&x);

int a[100];

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

{

printf("Enter the age of %d: ", (i+1));

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

}

sort(a,x);

printf("\nAfter sorting: ");

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

printf("\n%d",a[i]);

}

**Output**

Enter the number of employees: 5

Enter the age of 1: 32

Enter the age of 2: 23

Enter the age of 3: 53

Enter the age of 4: 23

Enter the age of 5: 64

Heap built:

23

23

53

32

64

Inserting… 23 23 53 32 64

Inserting… 23 32 53 64 23

Inserting… 32 64 53 23 23

Inserting… 64 53 32 23 23

Inserting… 64 53 32 23 23

**Alternative Code**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<limits.h>

int heap[10000000], size;

void ini()

{

size = 0;

heap[0] = -INT\_MAX;

}

void insert(int e)

{

size++;

heap[size] = e;

int n = size;

while(heap[n/2]>e)

{

heap[n] = heap[n/2];

n/=2;

}

heap[n] = e;

}

int dmin()

{

int min\_e = heap[1];

int last\_e = heap[size--];

int n;

int ch;

for(n=1;n\*2<=size;n=ch)

{

ch = n\*2;

if(ch!=size && heap[ch+1]<heap[ch])

ch=ch+1;

if(last\_e>heap[ch])

heap[n]=heap[ch];

else

break;

}

heap[n] = last\_e;

return min\_e;

}

void main()

{

int num;

printf("Number of elements: ");

scanf("%d", &num);

int ar[num];

int i, e;

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

{

printf("\nEnter element #%d: ",(i+1));

scanf("%d",&e);

insert(e);

}

for(i=(num-1);i>=0;i--)

ar[i] = dmin();

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

{

printf("\n%d",ar[i]);

}

}

Exercise - 3

**Question**

Create a binary tree and display its elements in pre-order, post-order, and in-order.

**Code**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* create()

{

int a;

struct Node\* n = (struct Node\*)malloc(sizeof(struct Node));

printf("Enter data, and -0 for no new Node: ");

scanf("%d",&a);

if(a==-0)

return 0;

n->data=a;

n->left = n->right = NULL;

printf("Enter left child of %d: ",a);

n->left = create();

printf("Enter right child of %d: ",a);

n->right = create();

return n;

}

void preorder(struct Node\* temp)

{

if(temp!=NULL)

{

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

preorder(temp->left);

preorder(temp->right);

}

}

void postorder(struct Node\* temp)

{

if(temp!=NULL)

{

postorder(temp->left);

postorder(temp->right);

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

}

}

void inorder(struct Node\* temp)

{

if(temp!=NULL)

{

inorder(temp->left);

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

inorder(temp->right);

}

}

void main()

{

struct Node\* root = create();

printf("\n\nPreorder: ");

preorder(root);

printf("\n\nPostorder: ");

postorder(root);

printf("\n\nInorder: ");

inorder(root);

printf("\n");

}

**Output**

Enter data, and -0 for no new Node: 1

Enter left child of 1: Enter data, and -0 for no new Node: 2

Enter left child of 2: Enter data, and -0 for no new Node: 4

Enter left child of 4: Enter data, and -0 for no new Node: -0

Enter right child of 4: Enter data, and -0 for no new Node: -0

Enter right child of 2: Enter data, and -0 for no new Node: 3

Enter left child of 3: Enter data, and -0 for no new Node: -0

Enter right child of 3: Enter data, and -0 for no new Node: -0

Enter right child of 1: Enter data, and -0 for no new Node: 5

Enter left child of 5: Enter data, and -0 for no new Node: 6

Enter left child of 6: Enter data, and -0 for no new Node: -0

Enter right child of 6: Enter data, and -0 for no new Node: -0

Enter right child of 5: Enter data, and -0 for no new Node: 8

Enter left child of 8: Enter data, and -0 for no new Node: -0

Enter right child of 8: Enter data, and -0 for no new Node: -0

Preorder: 1243568

Postorder: 4326851

Inorder: 4231658