WEEK 9 ADA LAB

1BM21CS247

q)IMPLEMENT N-QUEENS IN C USING BACKTRACKING METHOD:-

SOURCE CODE:-

#include<stdio.h>

#include<math.h>

int board[20],count;

int main()

{

int n,i,j;

void queen(int row,int n);

printf(" - N Queens Problem Using Backtracking -");

printf("\n\nEnter number of Queens:");

scanf("%d",&n);

queen(1,n);

return 0;

}

//function for printing the solution

void print(int n)

{

int i,j;

printf("\n\nSolution %d:\n\n",++count);

for(i=1;i<=n;++i)

printf("\t%d",i);

for(i=1;i<=n;++i)

{

printf("\n\n%d",i);

for(j=1;j<=n;++j) //for nxn board

{

if(board[i]==j)

printf("\tQ"); //queen at i,j position

else

printf("\t-"); //empty slot

}

}

}

/\*funtion to check conflicts

If no conflict for desired postion returns 1 otherwise returns 0\*/

int place(int row,int column)

{

int i;

for(i=1;i<=row-1;++i)

{

//checking column and digonal conflicts

if(board[i]==column)

return 0;

else

if(abs(board[i]-column)==abs(i-row))

return 0;

}

return 1; //no conflicts

}

//function to check for proper positioning of queen

void queen(int row,int n)

{

int column;

for(column=1;column<=n;++column)

{

if(place(row,column))

{

board[row]=column; //no conflicts so place queen

if(row==n) //dead end

print(n); //printing the board configuration

else //try queen with next position

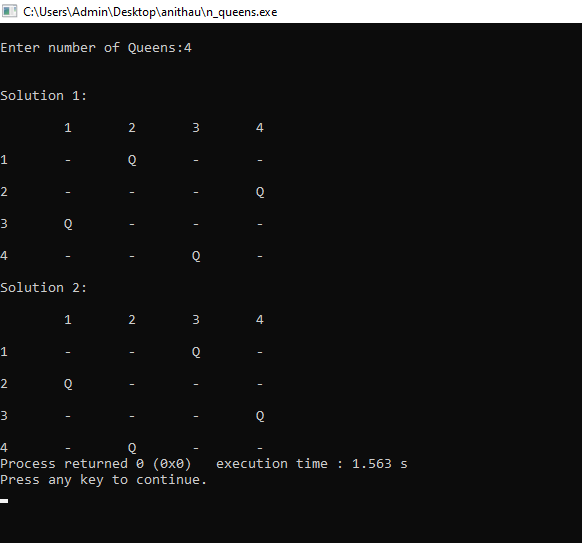
queen(row+1,n);

}

}

}

OUTPUT:-



Q)IMPLEMENT HEAP SORT USING A C PROGRAM

SOURCE CODE:-

// Heap Sort in C

#include <stdio.h>

// Function to swap the position of two elements

void swap(int\* a, int\* b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// To heapify a subtree rooted with node i

// which is an index in arr[].

// n is size of heap

void heapify(int arr[], int N, int i)

{

// Find largest among root,

// left child and right child

// Initialize largest as root

int largest = i;

// left = 2\*i + 1

int left = 2 \* i + 1;

// right = 2\*i + 2

int right = 2 \* i + 2;

// If left child is larger than root

if (left < N && arr[left] > arr[largest])

largest = left;

// If right child is larger than largest

// so far

if (right < N && arr[right] > arr[largest])

largest = right;

// Swap and continue heapifying

// if root is not largest

// If largest is not root

if (largest != i) {

swap(&arr[i], &arr[largest]);

// Recursively heapify the affected

// sub-tree

heapify(arr, N, largest);

}

}

// Main function to do heap sort

void heapSort(int arr[], int N)

{

// Build max heap

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

heapify(arr, N, i);

// Heap sort

for (int i = N - 1; i >= 0; i--) {

swap(&arr[0], &arr[i]);

// Heapify root element

// to get highest element at

// root again

heapify(arr, i, 0);

}

}

// A utility function to print array of size n

void printArray(int arr[], int N)

{

for (int i = 0; i < N; i++)

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

printf("\n");

}

// Driver's code

int main()

{

int arr[] = {-10, 9, -18, 7, 6, 1 };

int N = sizeof(arr) / sizeof(arr[0]);

// Function call

heapSort(arr, N);

printf("Sorted array is\n");

printArray(arr, N);

}

OUTPUT:-

