

WEEK 8

Y. Shamil Ahamed
1BM21CS248.

HEAPSORT:

INPUT:

```
#include<stdio.h>
void heap_adj(int a[],int n)
{
    int i,j,item;
    j=0;
    item=a[j];
    i=2*j+1;
    while(i<n)
    {
        if((i+1)<=n-1)
        {
            if(a[i]<a[i+1])
                i++;
        }
        if(item<a[i])
        {
            a[j]=a[i];
            j=i;
            i=2*j+1;
        }
        else
            break;
    }
    a[j]=item;
}

void heap_const(int a[],int n)
{
    int i,j,k,item;
    for(k=0;k<n;k++)
    {
        item=a[k];
        i=k;
        j=(i-1)/2;
        while(i>0 && item>a[j])
        {
            a[i]=a[j];
```

```

        i=j;
        j=(i-1)/2;
    }
    a[i]=item;
}
}
void heapsort(int a[],int n)
{
    int i,temp;
    heap_const(a,n);
    for(i=n-1;i>0;i--)
    {
        temp=a[i];
        a[i]=a[0];
        a[0]=temp;
        heap_adj(a,i);
    }
}
void main()
{
    int n,i;
    printf("Enter the number of elements:");
    scanf("%d",&n);
    int a[n];
    printf("Enter the elements:");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    heapsort(a,n);
    printf("After sorting:\n");
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
}

```

OUTPUT:

```

Enter the number of elements:7
Enter the elements:50 25 30 75 100 45 80
After sorting:
25      30      45      50      75      80      100

```

n-Queens:

INPUT:

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

```

Enter number of Queens:4

Solution 1:
      1      2      3      4
1      -      Q      -      -
2      -      -      -      Q
3      Q      -      -      -
4      -      -      Q      -

Solution 2:
      1      2      3      4
1      -      -      Q      -
2      Q      -      -      -
3      -      -      -      Q
4      -      Q      -      -

```