#include<stdio.h>

#include<stdlib.h>

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

{

int tmp = \*a; \*a = \*b; \*b = tmp;

}

void InsertionSortOneStep(int \*P, int i, int N)

{

int j;

int tmp = P[i];

for(j=i; j>0; j--)

{

if(tmp<P[j-1])

P[j] = P[j-1];

else

break;

}

P[j] = tmp;

}

void ScanNumber(int \*P, int N)

{

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

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

}

void CopyArr(int \*P1, int \*P2, int N)

{

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

P2[i] = P1[i];

}

int IsEqual(int \*P, int \*P1, int N)

{

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

{

if(P[i]!=P1[i])

return 0;

}

return 1;

}

void ShowResult(int \*P ,int N)

{

printf("%d", P[0]);

for(int i=1; i<N; i++)

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

}

void BuildHeap(int \*P, int N)

{

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

PerDown(P, i, N);

}

void PerDown(int \*P, int i, int N)

{

int Child;

int tmp = P[i];

for(; 2\*i+1<N; i=Child)

{

Child = 2\*i+1;

if(P[Child]<P[Child+1] && Child!=N-1)

Child++;

if(tmp<P[Child])

P[i] = P[Child];

else

break;

}

P[i] = tmp;

}

//void HeapSortOneStep(int \*P, int N);

int main()

{

int N;

scanf("%d", &N);

int \*P1 = (int \*)malloc(N\*sizeof(int));

int \*P2 = (int \*)malloc(N\*sizeof(int));

ScanNumber(P1, N);

CopyArr(P1, P2, N);

int \*CP = (int \*)malloc(N\*sizeof(int));

ScanNumber(CP, N);

//InsertionSort Test

int i;

int flag = 0;

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

{

InsertionSortOneStep(P1, i, N);

if(IsEqual(P1, CP, N))

{

flag = 1;

break;

}

}

if(flag == 1)

{

printf("Insertion Sort\n");

InsertionSortOneStep(P1, i+1, N);

ShowResult(P1 ,N);

}

else

{

BuildHeap(P2, N);

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

{

Swap(&P2[i], &P2[0]);

PerDown(P2, 0, i);

if(IsEqual(P2, CP, N))

{

printf("Heap Sort\n");

Swap(&P2[i-1], &P2[0]);

PerDown(P2, 0, i-1);

ShowResult(P2 ,N);

break;

}

}

}

return 0;

}