IIT PALAKKAD

COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology, Palakkad
CS2130: Data Structures and Algorithms Lab
Lab 3 (Min-Heap Using Arrays)

31 August, 2018

Time: 3 hrs

General Instructions

- You are expected to follow good programming practices (refer to the book titled *C How to Program* by Dietel and Dietel). Your programs should handle all kinds of test cases - positive, negative, corner cases/boundary conditions, good input, bad input and are never expected to crash on any input.
- You are also expected to provided a high level description of the functions used in your programs. There is no restriction on the format except that it should be concise and precise. Further, the running time and additional space (excluding the space for the arguments) used by the functions should be clearly stated and justified.
- **Please note that poor programming practices and improper documentation can attract a penalty of up to 20% of the marks allocated to a question.**

1 Part 1: Review of Sorting

1. Write a function that takes as its argument an integer array **A** of length n such that $\boldsymbol{A}[0] \geq \boldsymbol{A}[1] \geq ::: \geq \boldsymbol{A}[n-2]$. This function should shift all elements in array **A** that are lesser than $\boldsymbol{A}[n-1]$ to one position ahead of their current positions. This function should then place $\boldsymbol{A}[n-1]$ in the vacant position that arises due to this shift. Ensure that your function runs in $\mathcal{O}(n)$ time and uses only constant extra space (excluding the space for $\boldsymbol{A}$). Name the file as **11.c**. [15]

   Input Format: The input should be read from standard input. The first line contains the value of n and the subsequent lines contain the array elements itself with one element per line.

   ```
   Sample Input:
   5
   10
   8
   5
   1
   6
   ```

   Output Format: The output (displayed on the command line) should contain n lines with the ith line being the element $\boldsymbol{A}[i]$ in the resultant array $\boldsymbol{A}$.

   ```
   Expected Output:
   10
   ```

Continued on next page...

```
8
6
5
1
```

2. Write a program that takes two sorted arrays as input and merges them into a single sorted array. Ensure that the program runs in linear time. Name the file as **12.c**. [15]

   Input Format: The first line contains the size n of the first array. The n subsequent lines contain the array elements of the first array with one element per line. The next line contains the size m of the second array and the m subsequent lines contain the elements of the second array with one element per line.

```
Sample Input:
2
1
100
3
5
8
80
```

   Output Format: The output (displayed on the command line) should contain $m + n$ lines with the ith line being the ith element in the merged sorted array.

```
Expected Output:
1
5
8
80
100
```

2 Part 2: Heaps

1. Implement the standard operations of the min-heap data structure. A skeleton of the code is given and you should implement the functions that are left empty. Recall that a min-heap is a nearly complete binary tree with the property that the value of any node is at least the value of its parent. The nodes in the heap should have non-negative integral data and the running time of each of the operations must be $\mathcal{O}(\log n)$ where n is the number of elements in the heap. Name the file as **21.c**. [70]

```
int heapSize=0;
int parent(int i) { /* Fill in */ }
int left(int i) { /* Fill in */ }
int right(int i) { /* Fill in */ }
void swap(int *x, int *y){ /* Fill in */ }
int insertKey(int A[], int key)
```

Continued on next page...

```
{
/* Insert the element key into the heap represented by A.
Return 1 if the operation is successful and -1 otherwise. */
}
int deleteKey(int A[], int i)
{
/* Delete the element A[i] from the heap represented by A.
Return 1 if the operation is successful and -1 otherwise. */
}
int decreaseKey(int A[], int i, int newVal)
{
/* Decrease the value of A[i] to newVal. Return 1 if the
operation is successful and -1 otherwise. */
}
void minHeapify(int A[],int i)
{
/* Ensure that the subtree rooted at A[i] is a min heap. */
}
int extractMin(int A[])
{
/* Delete the root of the min heap represented by A. Return
the deleted element if the operation is successful and -1
otherwise. */
}
int getMin(int A[])
{
/* Get the root of the min heap represented by A. Return
the element if the operation is successful and -1 otherwise. */
}
void print(int A[])
{
/* Display the heap represented by A in the increasing order
of their indices, one element per line.*/
}
```

Input-Output Format: The input will be given in a file with the following format. The first line contains the value of n, the maximum size of the array A representing the heap. The subsequent lines will contain specific operations on the heap. The input-output format are as follows

INS x denotes **insertKey**($A; x$). Output **-1** if the operation was unsuccessful.
DEL i denotes **deleteKey**($A; i$)). Output **-1** if the operation was unsuccessful.
EXT denotes **extractMin**(A). Output should be the return value of this function.
PRT denotes **print**(A). Output should be the elements of the heap in the increasing order of their indices in the underlying array, one element per line.
DEC i x denotes **decreaseKey**(A, **i, x**). Output **-1** if the operation was unsuccessful.
MIN denotes **getMin**(A). Output should be the return value of this function.

Continued on next page...

```
Sample Input:
10
EXT
INS 7
INS 9
INS 60
INS 23
INS 5
PRT
DEL 1
PRT
DEC 2 10
PRT
EXT
PRT
MIN
PRT
INS 10
PRT
INS 15

Expected Output:
-1
5
7
60
23
9
5
9
60
23
7
9
60
23
7
9
60
23
7
9
60
23
10
-1
```