SOFTWARE ENGINEERING PROJECT
REPORT
GROUP 15

REQUIREMENT SPECIFICATION

In this project we are required to implement a software program that will compare the different sorting algorithms .The different sorting algorithms that we will be looking at are namely quick Sort(),heap Sort (),bubble Sort(),also the program has to do permute Array,gen array,and we should able to do test program().In a quick Sort() we are required to design a code that will sort the random number of integers .Our quick Sort() should sort numbers according to which the order the user wants,the sort can be ascending or descending.

In a heap sort we will be doing a code that will sort array of those numbers that were generated into a permuted values,program should build a array in a way that the largest value is at the root .In bubble sort our program has to repeatedly move through the list of array to be sorted and compare each pair of adjacent numbers and swap them if they are in the wrong order since they were generated randomly .In permute array our program have to rearrange numbers in an array in the same specified order in which it will be the user's preference .To permute any number in an array the user need to specify the permutations with an index array to point out the position of the elements in an array.

In print array the program has to return a systematic arrangement of numbers in a user readable form ,and that array should be shown in the screen for the user to see them .In gen array the program should generate new systematic arrangement by an algorithm that will returns a sequence of apparently numbers that are not related/non-related numbers each and every time the program is called .The test program should be available in our program so that we can be able to test our program that it works correctly with out giving errors if it does give errors or it does what we is not required by the user we can be able to make changes to ensure that it does is required to do.

We are also required to produce a software program that will generate some N=100 numbers of integers .The program has to generate those integers numbers randomly .The numbers generated must be between 1 and 65536 .Those random numbers we believe that they are really helpful in simulation and games where for example in a game of cards the program should generate different random cards numbers for a user to play with,since it will be useless or it wont be interesting to play different games with the same numbers .Then after the program has done generating those numbers it has to or should print them in a systematic arrangement way ,that is they should be generated into an array.

The program will just print or display the integers in an unsorted array .The program will print those integers lets say we want the program to generate ten(10) integers then we should invoke one of the sorting algorithms that we have mentioned above to sort those 10 integers in into a ascending or descending order depend on the user preference .Then after the code has sorted those integers it is required that they must be printed in the sorted and sequential array of those integers.

The program will be required that when the user invoke a permute array() on the sequential sorted sequence of generated integers that were printed by the program at first to scramble the content of the sorted array again .The program has to do the process again by printing the unsorted array of integers,then sort those integers and print the version the is sorted .For the second time the code should no invoke quick sort but it has to perform other one of the sorting algorithms mentioned in the

beginning lets say it performs heap sort.
The program can then repeat the same process using other sorting algorithm lets say using bubble sort() that will sort the integers by repeatedly stepping through the array list that need to be sorted,comparing each pair of adjacent integers and swapping them if they are in a wrong order .The process of bubble sort pass through the list of array repeatedly until no swaps are needed,which that will be a clearly indication that the list is correctly sorted .So after bubble sort the user instead of scrambling with permute array ,the user can then just generate a new set of random integers by invoking gen program().in this software program the user is required or expected to invoke the permute array() at least once .So it is also required that we manage individual function and test our program if it works correctly.

In part two of software program after we have done and the program is working/running without giving the user any errors,we are required to run same timing of our code with our N being 10000 this time .After changing our N to 10000 the program should not print any output of array on the screen .So then we are required make some changes to our test program to include some timing .The timing that we have to produce has to tells us how much time the program takes to sort 10000 integers by using those three different algorithms that are quick sort ,heap sort,bubble sort .The results of the time that the program takes to sort should be bring into existence in a pleasing table.

In the third part of our program we are required to develop and include some additional components(modules) to the tasks that we have done above,so that data that is in the form of records of key/value pairs one in a line can read from an input text file and stored in a GNU implementation of the standard Unix *dbm* library namely *gdbm* data store. Those records will be subsequently read from the gdbm data store that is written into an output file in a sorted order of their keys. The order in which the records can be either ascending or descending according to a flag (ASC,DESC) that is passed to the separable component that does the sort.

So we will be given a file of records that will be consisting of key/value pairs. The key will be an integer number consisting of six digits and their associated value, which will be the last name and first name. Then in each record we will be having three fields separated by a commas. So we will be required to take those records and insert them into a gdbm data store. When we are done with reading the input file and creating gdbm data store, our program will have to print out the first 15 records and the last 15 records in the gdbm data store, each record should be printed out in its line. When we have finished generating the output file of the records sorted on their associated keys, our code has to print out again 15 records, the first one's and the 15 records the last one's of the lab3output.txt each record has to be in its line.

The acceptance criteria are:

- the user can key in gen Array and the random integers generated will be shown
- the user has to use the command or the terminal to enter and get the output.

The user has an input flag (ASC,DESC) to enter and get output the manner the user wants.

DESIGN AND IMPLEMENTATION

The pseudo codes for our different programs to guide us to the program with a clearly and enough information or details. Pseudo code will allow us to focus on the logic of algorithm that will solve the problem without being destructed by details,lets the following pseudo codes given as:

A BUBBLE SORT ALGORITHM:

```
function bubble sort[int ar[],int size];
begin;
        generate outputs array ar of size n;

        local variables j and temp;

        for(int j equal zero,i<size,increment i);

        for(int j equal zero,i<size -j-1,increment j);
        if (ar[j] greater than ar[j+1]);

                temp equal ar[j];

                ar[j] equal ar[j+1];

                ar[j+1] equal temp;

        end for loop;
  end for loop;
end
```

HEAP SORT ALGORITHM:

```
function heap sort ( i,size);

function build heap ( i,size);
function percolate dawn (int heap[] ,int size,int id);

generate and unordered array I of length size;

function heap sort(array[][,int size);

        build heap(array,size);
        int heap size equal size;
        for(int I equal size -1,i greater than zero,decrement i);
        int temp equal array[i];
        array[i] equal array[0];
        array[0] equal temp;
        heap size decrement;
percolate dawn(array,heap size,zero);


function build heap ( int array[],int size);
for(int I equal size/2,i greater or equal zero,decrement i);

        int current equal id;
```

```
        int max;

        while true;
        int left equal current *2+1;
        int right equal current *2+1;
        if left greater than size;
        return;

        else if right greater than size;
        print max equal left;

  if array[left] less than array[right];
        max equal right;

    if array[left] greater than array[right];
        max equal left;

if array[max] greater than array[current];

    int temp equal array[max];
   array[max] equal array[current];

array current equal temp;
current equal max
else
return
end
```

<u>THE PSEUDOCODE FOR THE MAIN PROGRAM IS AS FOLLOWS;</u>

```
function main()

cout “please enter the size of an array”;
        enter int size;
        print size of an array;
int arrayTosort [size];
for int i equal zero ,i less than size,increment i;
arrayTosort[i] equal I + rand() 10000;
cout “the unsorted array is”;
print Array(arrayTosort,size);
        cout \n;

  cout “the sorted array using the heap sort becomes ”;
        heap sort(arrayTosort,size);
print Array(arrayTosort,size);

random-shuffle arrayTosort[0], arrayTosort[size];
```

```
        cout “before sorting” \n;
         print array(arrayTosort,size)
    sort(arrayTosort,size);
cout “after bubble sorting ” \n;

print array(arrayTosort,size)

end
```

PRINT ARRAY PSUEDOCODE

```
function print array(integer array,integer size);
constant integer v set to be 10;

      for(integer i equal zero,i less than size,increment i)
          if(i equal v and v less than size)
                cout “\n v equal v incremented by 10”
         cout “array[i]”
cout;
```