Shiv Nadar University

CSD101: Introduction to Computing and Programming

Endsem Re-Exam

Max marks: 100 15-1-2022

Time: 135 mins for exam + 15mins (for upload, internet/power problems etc.).
Submit by 5.30pm.

1. *Answer all 3 questions.*
2. *For all program fragments assume that they are embedded in programs that compile and link without errors.*
3. *Answer each question directly on Blackboard. PDF/image files are not acceptable.*
4. *I will be available in the online classroom EndsemReExam on BB during the exam for any clarifications. You can enter the classroom and ask your questions orally.*
5. *A submission after 5.30pm will not be graded. This is a hard deadline*

1. The median element in an array $arr[N]$ is the middle value element. That is it has an equal number of elements that are less than or equal to the element and that are greater than or equal to it. For example if the array is `arr=[3,5,1,6,2,8,7]` then `5` is the median element since `1,2,3` are less than or equal to `5` and `6,7,8` are greater than or equal to `5`. For simplicity we assume that the size of the array is always odd.

   The code below, which has some missing parts, finds the median element by repeatedly finding the maximum element in the relevant part of the array and putting it at a suitable place in the array.

   The number of elements in the array and the element values are read from the file `input.txt`. Note that the number of elements should always be odd. The median element is printed on the terminal.

   Answer the questions given based on the code and its behaviour.

```
#include<stdio.h>
#include<stdlib.h>
#define N 50

int findMaxIndex(float arr[], int start, int end) {
   /*Finds and returns the index of the largest element between
   indices start and end, both inclusive.
   */
   int maxInd=?1?;//initializes index of max element
   for (int i=?2?; ?3?; i++)
      if (?4?) maxInd=i;
   return maxInd;
```

```
}

float median(float arr[], int size) {
   /*Finds and returns the median element in the
   array arr, size is always odd.
   */
   int mid=?5?, maxind;
   float tmp;
   for (int j=0; ?6?; j++) {
      maxind=?7?;
      //Swap elements between maxind and j
      ?8?
   }
   return ?9?;
}

int main() {
   float arr[N];
   int size;
   FILE *inFile=?10?;
   if (?11?) {printf("File not found.\n"); exit(0);}
   fscanf(?12?);//reads number of elements
   //Loop to read elements into the array
   for (int i=0; i<size; i++)
      fscanf(?13?);
   printf("Median = %f\n",?14?);
   exit(0);
}
```

(a) Fill in the missing code fragments `?1?` to `?14?` so that the program works correctly. Comments are present in the program to help you.

You do not have to write the full program. Just give the serial number and the corresponding code fragment.

Solution:

1 $start$
2 $maxInd + 1$ or $maxInd$ or $start + 1$ or $start$
3 $i <= end$
4 $arr[i] > arr[maxInd]$
5 $mid = size/2$
6 $j <= mid$
7 $findMaxIndex(arr, j, size - 1)$
8 $tmp = arr[j]$ $arr[j] = arr[maxind]$; $arr[maxind] = tmp$;

```
9   arr[mid]
10   fopen("input.txt","r")
11   inFile == NULL
12   inFile,"%d",&size
13   inFile,"%f",&arr[i]
14   median(arr,size)
```

(b) Based on the program what is the $O(.)$ (big O) complexity of finding the median? Justify your answer. No marks without proper justification.

Solution:
$O(n^2)$. Function `findMax` takes $O(m)$ where m is the length of the array in each round namely, $n, n-1, n-2$ etc. in successive rounds and this is done $\frac{n}{2}$ times so $O(n^2)$.

(c) If the file `input.txt` has the following content what will be the output.

```
9
1.2 4.5 17.2 8.6 3.4 2.9 10.0 88.3 0.9
```

Solution:
4.5

(d) For the file `input.txt` above what will be the contents of array `arr` just before the program exits. Show only the relevant contents.

Solution:
`88.3 17.2 10.0 8.6 4.5 2.9 3.4 1.2 0.9` - actually the relevant part is only the first 5 elements which will be in decreasing order.

(e) Briefly outline another algorithm that will solve the same problem a little more efficiently than the algorithm in the code above. What will be the complexity of this more efficient algorithm?

Solution:
Merge sort the array in $O(n\,log(n))$ time and then pick the middle element. This is better than $O(n^2)$.

[2×14,8,2,4,8=50]

2. The following three questions pertain to a **doubly linked list**.

(a) Write the structure definition for a node of a doubly linked list where the data field contains a single character as data, the forward pointer is named `after` and the backward pointer is named `before`.

Solution:

```
struct Node {
    char data;
    struct Node *before, *after;
```

(b) Assume the variable `dllp` points to a node in the doubly linked list and we want to insert a node with a data value of 'z' immediately after the node pointed to by `dllp`. Write the C code fragment along with any necessary declarations that will do this.

Solution:

```
//Create memory for the new node
struct Node *new=malloc(sizeof(struct Node));
new->data='z';//set data field
//If dllp is NULL list is empty
if (dllp==NULL) {new->after=NULL; new->before=NULL; dllp=new;}
else {//dllp is not NULL
    new->after=dllp->after;
    new->before=dllp;
    dllp->after=new;
}
```

(c) As in the previous part assume `dllp` is pointing to a node in the doubly linked list and we want to delete that node. Write the C code fragment along with any necessary declarations that will accomplish this.

Solution:

```
//Delete only if not NULL
if (dllp!=NULL) {
    (dllp->before)->after=dllp->after;
    (dllp->after)->before=dllp->before;
    free(dllp);
}
```

[6,12,12=30]

3. You have 3 C program files called `p1.c`, `p2.c` and `p3.c`.The files `p1.c` and `p2.c` contain function definitions that are used in the file `p3.c` which also contains the `main` function. In addition the file `p2.c` uses functions defined in `p1.c`.

(a) Write `gcc` command(s) that will allow you to generate the object files `p1.o`, `p2.o` and `p3.o`.

Solution:

gcc -c p1.c p2.c p3.c

(b) Write the `gcc` command that will generate an executable named `myexe` from the object files generated previously.

Solution:
gcc -o myexe p3.o p1.o p2.o

(c) In addition to any standard header files what other header files will be present in `p3.c` and `p2.c`?

Solution:
In p3.c `#include "p2.h"` and `#include "p1.h"`
In p2.c `#include "p1.h"`

(d) Write a `makefile` that allows one to do the actions in parts (a) and (b) above.

Solution:
The makefile is shown below:

```
myexe : p3.o p1.o p2.o
    gcc -o myexe p3.o p1.o p2.o

p3.o : p3.c p2.o p1.o p2.h p1.h
    gcc -c p3.c

p2.o : p2.c p1.o p1.h
    gcc -c p2.c

p1.o : p1.c
    gcc -c p1.c
```

[6,3,4,7=20]