Data Structures

Time: 90 Minutes | Midterm – June 29, 2020

Total Marks: 50

Important! Read all the instructions.

- This exam has three parts.
- Each part has specific time.
- Each part has to be solved and submitted in its specified time.
- You can submit the paper through MS Teams or WhatsApp.
- Be honest! Do the exam yourself! Any kind of plagiarism will result in negative marking.
- If you have read the instructions, draw a star at the top right corner of the first page. You'll get 1 mark for it.

Part 1 – (20 minutes)

Q1: In this question, you will create a few random numbers, these numbers will be used in the subsequent questions.

a. What is your registration number? {2}

b. `N1 = last two digits of your registration number`

c. `N2 = 100 – N1`

d. `N3 = floor(N2 / 2)`

e. `N4 = N2 * 2`

f. `N5 = N2 + N3`

Q2: Make sure you have solved Q1 before attempting this questions. What would be the output of the following code? Show all the recursive calls in form of a tree. {8}

```
int h(int y) {
  if (y < 10)
    return y*2;
  else
    return h(y/4)+h(y/8);
}

int main() {
  //use the value of N4 from Question # 1
  cout<<h(N4)<<endl;
  return 0;
}
```

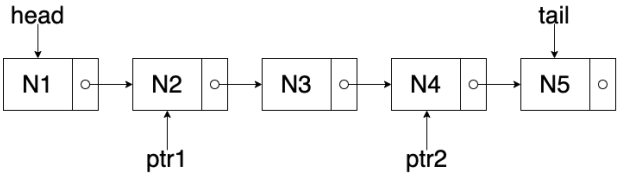
Part 2 – (30 minutes)

Q3: Convert the following expression into postfix notation. Do not use the variables, rather use the values of variables as defined in Question # 1. {6+4=10}

a. `(N1 + N4 – N2) * N1 * N3 / N5` Use the values of the variables as defined in Q1

b. Evaluate the postfix expression obtained in part (a)

Q4: Write down the results of the following expressions using the figure given below. In case of an error, mention it. {0.5+1+1.5+1.5+1.5+4=10}



a. Replace the variables N1 to N5 with their values as calculated in Q1 and draw the list again. Answer the subsequent parts based on the new list you draw.

b. `head->next->data`

c. `ptr1->next->next->data == ptr2`

d. `ptr2->data + tail->data`

e. `ptr2->next->next->data`

f.
```
int x = 0;
while (head != ptr2->next)
{
        x = x + head->data;
        head = head->next->next;
}
```

Part 3 – (20 minutes)

Q5: Implement a **client code** function *Common*, which takes two sorted lists as input and returns a sorted list containing the common values found in both the lists. The signature of the function should look like:

```
SortedList Common(SortedList &s1, SortedList &s2);
```

{20}

Extra Time – (20 minutes)

If some of your questions were left incomplete or you want to review them, you can do that and submit by the end of this extra time. Please note that whatever you do is an extension of the previous work and not a completely new work.