**Quiz Mid – Data Structures**

**Instruction:**

* Try to not look at notes, slides, or the web as much as possible. Consider this quiz as your try out for Mid Exam.
* Looking at notes, slides and the web is still possible if you are really stuck, however give the sources.
* Do the quiz alone without asking or discussing with your friend.
* For number 1, you may use excel or word (table). For other numbers, just submit the .cpp (total 3 .cpp files for each number). Zip them all, rename with format NIM\_NAME\_QUIZ1 and submit to forum.
* 10% of the points you get here will be allocated to bonus points for Mid Exam, ex: You get 70, I will add 7 points for your Mid Exam.
  + - 1. [20 points] Given this infix notation below:

1. Convert infix notation above to prefix notation. Simulate it using Stack Algorithm
2. Evaluate the prefix notation from (a) to find the calculation result. Simulate it using Stack Algorithm
   * + 1. [20 points] Given this code of Binary Search Tree below, fill the code as the instruction says:

struct Node {

int data;

struct Node \*left;

struct Node \*right;

};

int countNodes(struct Node \*root) { // This function will return the total number of nodes in the BST

if (root == NULL) {

return 0;

}

return (/\*fill this line of code\*/);

}

int findHeight(struct node\* node){ // This function will return the height of the tree

if (node == NULL)

return 0;

else {

/\* fill the lines of code here \*/

if (leftHeight > rightHeight)

return (leftHeight + 1);

else

return (rightHeight + 1);

}

}

void search(struct Node \*current, int find) {

if (/\*fill this line of code\*/) {

printf("%d is not found\n", find);

} else if (/\*fill this line of code\*/) {

printf("%d is found\n", current->data);

} else if (/\*fill this line of code\*/) {

/\*fill this line of code\*/

} else {

search(current->right, find);

}

}

* + - 1. [20 points] Given this code of hashing with linear probing below, fill the code as the instruction says:

#define SIZE 10

struct Node{

char name[20];

}\*Array[SIZE];

int count = 0;

struct Node\* newNode(char name){

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

strcpy(newNode->name,name);

return newNode;

}

void insert(char name[]){

int hashkey = hashFunction(name);

int point = hashkey;

if(/\*fill this line of code\*/){

printf("The Array is Full");

return;

} else {

while(Array[point] != NULL){

/\*fill the lines of code\*/

}

}

Array[point] = newNode(name);

}

void delete(char name[]){

int hashkey = hashFunction(name);

int point = hashkey;

do{

if(Array[point] && strcmp(Array[point]->name,name) == 0){

/\*fill the lines of code\*/

return;

}

/\*fill the lines of code\*/

}while(point != hashkey);

return;

}

* + - 1. [40 points] Make Priority Queue – Double Linked List (Insert, View, Search, Delete)
         1. Create a task list application consist of task id and priority level.
         2. Priority level of task will be between “HIGH” and “LOW”.
         3. HIGH priority task must be done first, and LOW priority task can be done later.
         4. The application will receive 3 inputs: INSERT A1 A2, VIEW, DO, and DOALL where A1 is the id (consisting of 4 numbers) and A2 is the priority level (HIGH/LOW).
         5. INSERT will insert an id (consisting of 4 numbers) and the priority level (HIGH/LOW). Don’t forget to validate.
         6. VIEW will view the current queue, including the Number, the Id and the Priority.
         7. DO will pop 1 task in the priority queue and print “[Id] is DONE. [count] items remaining…” where [count] is the number of remaining tasks in the queue.
         8. DOALL will pop all task one by one from the top of the queue, everytime it pops 1 tasks, it will print “[Id] is DONE.” Until the last tasks. At the end, print “[count] tasks remaining”.

Sample:

|  |  |
| --- | --- |
| Input | Output |
| INSERT 1111 HIGH  INSERT 2222 LOW  INSERT 3333 LOW  INSERT 4444 HIGH  INSERT 5555 LOW  VIEW  DO  DO  VIEW  DOALL | Task lists:  1 1111 HIGH  2 4444 HIGH  3 2222 LOW  4 3333 LOW  5 5555 LOW  1111 is DONE.  4 tasks remaining…  4444 is DONE.  3 tasks remaining…  Task lists:  1 2222 LOW  2 3333 LOW  3 5555 LOW  2222 is DONE.  3333 is DONE.  5555 is DONE.  0 tasks remaining |