BBM201 – Data Structures – Fall 2015
2nd Midterm
30.11.2015 – 9:30-11:20

Name Surname: _____

Student ID : _____ **Section:**_____

Duration: 100 minutes

Question	1	2	3	4	5	6	7	8	9	10	*	Total
Points	10	9	10	7	12	8	12	12	10	10	10	100
Grade												

Question 1. Decide if the following statements are true or false, circle your answer. Give a short explanation if you chose 'False'.

a) The worst-case cost to insert an element to the beginning of a linked list of size n is O(1).

True □ False □

b) The worst-case cost to insert an element to the end of an array of size n is O(n), if there is still space at the end of the array.

True □ **False** □

It is O(1) because there is no shifting. We only add the item to the given address.

c) The worst-case cost to delete an element from a linked list of size n is O(1).

True □ **False** □

It is O(n) because first we need to find the item to be deleted by traversing the linked list.

d) The worst-case cost to delete an element from an array of size n is O(n).

True □ False □

e) The memory used to store a linked list of 5 integers is 40 bytes (suppose each pointer is 4 bytes).

True □ False □

Question 2. A palindromic word is a sequence of characters that reads the same backward and forward. For example; *pepper*, *refer*, *kayak* are palindromic words. The method IsPalindrome(char[] str) checks if a given string is palindrome by using a stack. According to the given prototypes of stack methods, please fill in the gaps in the code.

```
int stack[10];
int top=-1;
void push(char); //pushes the given char on the stack.
char pop();      //pops a char from the stack.

void IsPalindrome(char str[]){

      int len = strlen(str), i, count=0;
      len = strlen(str);

      for (i = 0; i < len; i++)
          push(str[i]);

      for (i = 0; i < len; i++)
          if (str[i] == pop())
              count++;

      if (count==len)
          printf("\n%s is a Palindrome string\n", str);
      else
          printf("\n%s is not a palindrome string\n", str);

}
```

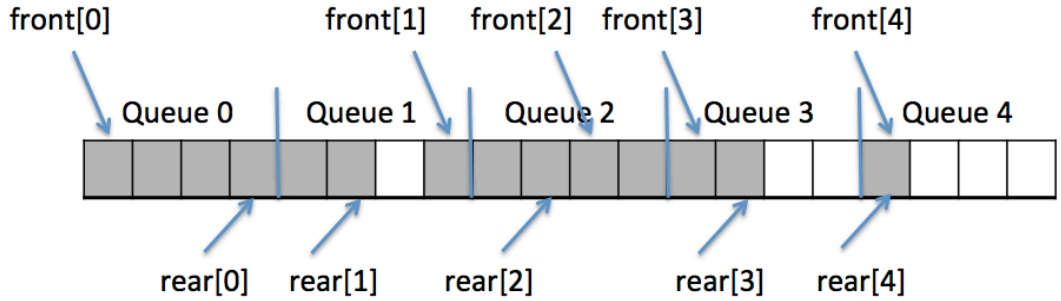
Question 3. Postfix is preferable than infix for computers because of having no checks for parenthesis and no check for operator precedence rules. Prefix has also the same properties as postfix for having no parenthesis and precedence check. Give a reason why postfix is more preferable than prefix for evaluation of expressions by the computers?

We can evaluate a postfix notation by using only one stack, whereas in prefix notation we have to use at least two stacks for the evaluation. In addition, we need to traverse to the end of the prefix notation and then we need to calculate the expression backwards.

Question 4. Let `48+2*35-+47-*` be an expression given in postfix notation (suppose each number has only one digit again). Fill the table on the right accordingly, if the input is stored and processed using a stack notation in order to evaluate the given expression. ([0],[1],[2] indicate the position number of an element, [0] being the bottom position.)

Token	Stack			Top
	[0]	[1]	[2]	
4	4			0
8	4	8		1
+	4+8			0
2	4+8	2		1
*	(4+8)*2			0
3	(4+8)*2	3		1
5	(4+8)*2	3	5	2
-	(4+8)*2	3-5		1
+	((4+8)*2)+(3-5)			0
4	((4+8)*2)+(3-5)	4		1
7	((4+8)*2)+(3-5)	4	7	2
-	((4+8)*2)+(3-5)	4-7		1
*	(((4+8)*2)+(3-5))*(4-7)=-66			0

Question 5. Multiple stacks/queues have a recovery mode if it is full and needs to add more items. The main idea is finding another stack/queue that has a space, which we can use by shifting the items. A sample **multiple circular queue** is given below where there are 5 **circular queues** of size 4. Queue 0 and Queue 2 are full and others are not. For each queue a front and a rear pointer are kept.



Explain the steps that need to be taken when adding to a full queue and take recovery steps. Bear in mind that there are different situations of a full queue. Use short sentences. No coding is required.

If the rear comes after the front look at the right of the full queue in the multiple queue structure. If there is any space in any queue on the right, shift the items one space to the right. If the front comes after the rear, check if there is any space in any queue on the left, shift the items one space to the left to have one space for the new item.

Question 6. The expression "`14312+*/-352-*+2/`" is written in postfix notation. Write the expression in the following notations. Keep in mind that each number consists of only one digit.

(a) Infix notation:

((1-(4/(3*(1+2))))+(3*(5-2)))/2

(b) Prefix notation:

/+-1/4*3+12*3-522

Question 7. According to the given array based linked list which is defined as follows:

```
#define MAX_LIST 10

typedef struct{
        char letter;
        int link;
     }item;
item linkedlist[MAX_LIST];
int free_; //the index of the first free space in the list
int* list; //the index of the first item in the list
```

Imagine we begin with a sorted list given in the first table given below. After we insert "C" and delete "A" in the sorted list, what will be the contents of the list? Please fill in the tables and write down the values of free_ and *list.

Initial table

	letter	link
[0]	D	1
[1]	E	2
[2]	F	5
[3]	A	4
[4]	B	0
[5]	G	6
[6]	I	7
[7]	K	-1
[8]		9
[9]		-1

free_= 8
*list = 3

Insert "C"

	letter	link
[0]	D	1
[1]	E	2
[2]	F	5
[3]	A	4
[4]	B	8
[5]	G	6
[6]	I	7
[7]	K	-1
[8]	C	0
[9]		-1

free_= 9
*list = 3

Delete "A"

	letter	link
[0]	D	1
[1]	E	2
[2]	F	5
[3]		9
[4]	B	8
[5]	G	6
[6]	I	7
[7]	K	-1
[8]	C	0
[9]		-1

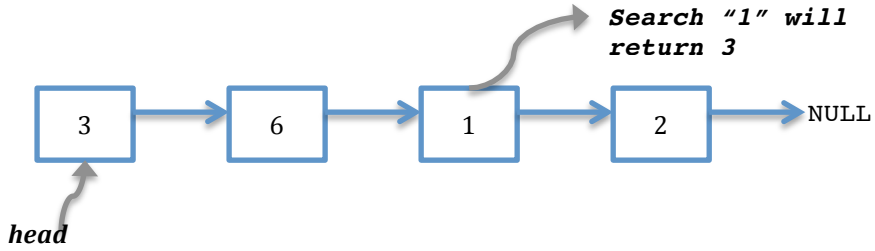
free_= 3
*list = 4

Question 8. Given an unsorted linked list, and without using any extra memory, fill in the blanks in the below method that will delete any duplicates from the linked list. (Brute force solution is acceptable.)

```
void RemoveDuplicates (struct Node * head)
{
    struct Node * temp1, * temp2, * prev;
    temp1=head;
    while(temp1!=null)
    {
        temp2=temp1->next;
        prev = temp1;
        while(temp2!=null)
        {
            if(temp1->data == temp2->data)
            {
                //delete
                prev->next = temp2->next;
                free(temp2);
                temp2 = temp2->next;
            }
            else
            {
                prev = temp2;
                temp2=temp2->next;
            }
        }
        temp1=temp1->next;
    }
}
```

Question 9.

Write a **recursive** method that searches for a value in a linked list and returns the position of the value in the linked list if found, otherwise returns -1. The first item in the linked list has position 1. You can add more parameters for the given method, if needed. (Using global variable is **not** allowed!)




```
struct Node{
       int data;
       struct Node* next;
};


int  SearchItem(struct Node* head, int value, int index)
{


      if(head==null)
            return -1;
      else if(head->data==value)
            return count+1;
      else
            SearchItem(head->next,value,index+1)
}
```

Question 10. Print the contents of the linked list returned by the Func function for the provided two linked lists and the code below.

list1-> 1 -> 3 -> 4 -> 6-> NULL
list2-> 0 -> 2 -> 5-> 7-> NULL
list3 = Func (list1, list2);

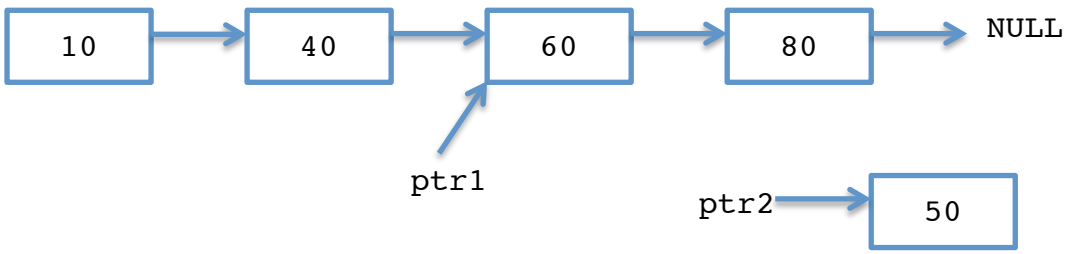
list3-> 0→1→2→3→4→5→6→7→NULL

```
Node* Func (Node *list1, Node *list2) {
      if (list1 == null) return list2;
      if (list2 == null) return list1;

      Node* newhead,temp;
      if (list1->data < list2->data) {
            newhead = list1;
            list1= list1->next;
      }
      else {
            newhead = list2;
            list2 = list2->next;
      }
      temp = newhead;

      while(list1!= null && list2 != null) {
            if (list1->data > list2->data){
                  temp->next = list2;
                  list2 = list2->next;
            }
            else {
                  temp->next = list1;
                  list1 = list1->next;
            }
            temp = temp->next;
      }
      if (list1 == null)
            temp->next = list2;
      else
            temp->next = list1;
      return newhead;
}
```

Bonus Question. Consider the sorted linked list with these nodes:



The only pointer that is given for this list is ptr1 that is pointing to the node with the value 60 (shown above) and there is no head pointer provided. There is also a new node with the value 50 and a pointer, ptr2, pointing to it.

Write the steps for inserting the new node to the above list in the sorted order. Explain your answer in sentences. No coding is required.

Insert 50 after 60, and replace their data to have 50→60→80…