BBM201 – Data Structures – Fall 2015

1st Midterm
26.10.2015

Ad Soyad / Name Surname: _____

Öğrenci No / Student ID : _____ **Şube / Section:** _____

Süre … dakikadır.

Question	1	2	3	4	5	6	7	8	Total
Points									100
Grade									

Question 1.
Mark the following statements as True or False. Correct the false statements with a concise explanation.

(a) A data structure is a way to store and organize data in computer, so that it can be used easily.

(b) Bottom-up and top-down analyses are two approaches in the system's life cycle.

(c) Big-Oh (O) notation gives a lower bound on the running time of a program.

(d) 3 n^2 + 10 n log n = Omega(n^2)

(e) 3 n^2 + 10 n log n = O(n log n)

Question 2.
Let an array definition be:

```
int a[5][5][5][5];
```

What is the memory address of a[3][4][2][1] if the memory address of a[0][0][0][0] is x?

Question 3.
Let a[n][n] be an upper triangular matrix (see the example given below). The elements of this triangular matrix are stored in a one-dimensional array as given below:

$$\begin{matrix} a_{oo} & a_{o1} & a_{o2} & a_{o3} \\ 0 & a_{11} & a_{12} & a_{13} \\ 0 & 0 & a_{22} & a_{23} \\ 0 & 0 & 0 & a_{33} \end{matrix}$$

U	a_{00}	a_{10}	a_{11}	a_{20}	a_{21}	a_{22}	a_{30}	a_{31}	a_{32}	a_{33}

Please complete the method readtriangularmatrix(int[], int) that reads integers from the keyboard and fills the one-dimensional array U with an upper triangular matrix.

```
void readtriangularmatrix(int U[], int n){

    int i, j, k;
    if(n*(n+1)/2 > MAX_SIZE){
       printf("\n invalid array size \n");
       exit(-1);
    }
    else
      for(i=0; i<=n-1; i++){
         k=……..
         for(j=0; j<=i; j++)
              scanf("%d", ……..);
      }
}
```

Question 4.

Please write the output of the following method.

```
void doSomething(int value)
{
      if(0 < value && value < 10)
      {
              doSomething(value - 1);
              doSomething(value + 1);
              printf(" %d", value);
      }
}
```

Question 5.

Please write the output of the following method.

```
int result = negative(-3);
int negative(int num)
{
      if(num >= 20)
           return -5;
      else
           return negative(num + 4) + 2 * num;
}
void print()
{
      printf("The final answer is %d", result);
}
```

BBM 201 - Data Structures - Fall 2015
Midterm 1
Date: October 26, 2015
Time: 9:30-11:20

Ad Soyad / Name:
Ögrenci No /Student ID:
Şube /Section:

Question	1	2	3	4	5	6	7	8	9	10	11	Total
Points	10	8	10	6	10	6	10	8	15	9	8	100
Grade												

1. (10 points) Decide if the following statements are true or false, circle your answer.

The cost of insertion (Push) and deletion (Pop) of an element in a stack is O(1). (True) False

The ~~complexity~~ order of growth of the worst-case running time of the 2-Sum problem for an input of size n is $n \log n$. True (False)

The ~~complexity~~ order of growth of the worst-case running time of the iterative Fibonacci algorithm for finding f_n is n. (True) False

The ~~complexity~~ order of growth of the worst-case running time of binary search for an array of size n is n^2. True (False)

$3n^2 + 10n \log n = O(n \log n)$ True (False)

2. (8 points)

(a) (4 points) Find what RecursiveFunc (given below) does **in general**.

Finds the maximum of elements in a

(b) (4 points) Print the output of the following program.

```
int RecursiveFunc (int array[], int index, int n)
{
     int val1, val2;
     if ( n==1 )
          return array[index];
     val1 = RecursiveFunc (array, index, n/2);
     val2 = RecursiveFunc (array, index+(n/2), n-(n/2));
     if (val1 > val2)
          return val1;
     else
          return val2;
}
...
int a[8] = 1,2,10,15,16,4,8,2;
printf( "value is % d \n", RecursiveFunc(a,0,8));
```

value is 16

3. (10 points) Below is a pseudocode of the Euclidean algorithm that calculates the greatest common divisor (gcd) of any given two integers. Write all recursive calls made to calculate gcd(21, 1039) by using a *box diagram*.

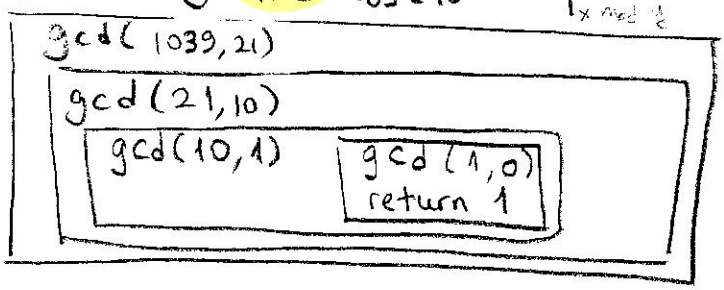
```
algorithm gcd(x,y)
  if y = 0
    then return(x)
  else return(gcd(y,x mod y))
```

x = 1039, y = 21

1039 | 21 = 49
- 84
199 − 189 = 10 (x mod y)

21 | 10 = 2
20
1 (x mod y)

10 | 1 = 10
10
0 (x mod y)



4. (6 points) Write the complexity of the running time of the following code fragment using big-Oh notation. Show your work.

```
int count = N, i = 0, j = 0;
for (i = 0; i<count; i++)
     if(i%2==0) // i , even
          j++;
     else
          j--;
for (i = count; i > 0; i = i/2)
     for (j = 0; j < i; j++)
          count ++;
```

$$\left\lceil \frac{N}{2} \right\rceil + \left\lceil \frac{N}{2} \right\rceil + N + \frac{N}{2} + \frac{N}{4} + \cdots + \frac{N}{2^k} =$$

$k = \log_2 N \qquad = O(N)$

5. (10 points) How many array accesses do the following code fragments make as a function of N?

(a) (5 points)

```
int count = 0;
for (int i = 0; i < N; i++)
     for (int j = i+1; j < N; j++)
          if (a[i] + a[j] == 0)
               count++;
```

$2 \cdot [(N-1) + (N-2) + \cdots + 1] = 2\binom{N}{2} = \frac{N(N-1)}{2}$

(b) (5 points)

```
int count = 0;
for (int i = 0; i < N; i++)
     for (int j = i+1; j < N; j = 2*j)
          if (a[i] + a[j] == 0)
               count++;
```

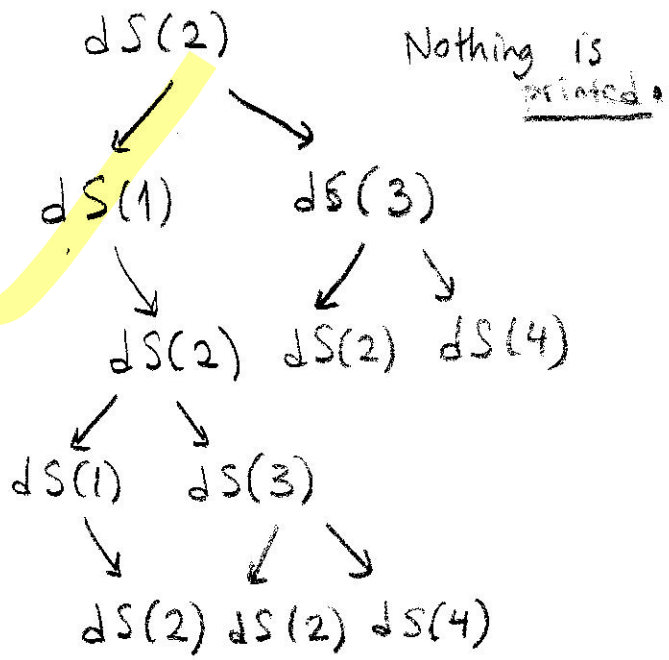
N times

$2\left[\log_2(N-1) + \log_2(N-1) + \cdots + \log_2(N-1)\right]$ ($i = 0$ to $i = N-1$)

$\sim 2N \log_2 N$

6. (6 points) What will be the output when doSomething(2) is called?

```
void doSomething(int value)
{
     if(0 < value && value < 10)
     {
          doSomething(value - 1);
          doSomething(value + 1);
          printf(" %d", value);
     }
}
```

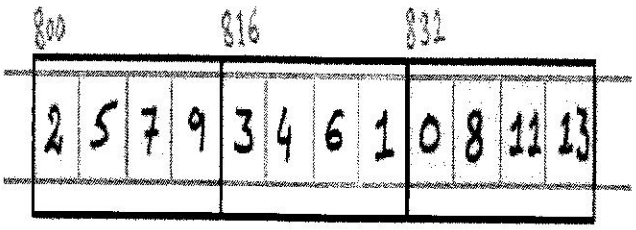


Nothing is printed.

Never Stops

7. (10 points) If C is the array shown with its address above each node, write what the following lines of a program will print in the empty column.

```
int C[3][2][2];
```



The code	*will print:*
`printf("C=%d", C);`	800
`printf("C+1=%d", C+1);`	816
`printf("*(*C+1)) =%d", *(*C+1));`	808
`printf(" *(C[1]+1)=%d '', *(C[1]+1)) ;`	824
`printf(" *(C[1][1]+1)=%d '', *(C[1][1]+1));`	1

8. (8 points) Let an array be defined with `int a[6][8][5][7]`. What is the memory address of `a[3][4][2][1]` if the memory address of `a[0][0][0][0]` is x? (Assume that size of an integer is 4 bytes.)

$$x + \left[3 \cdot (8 \cdot 5 \cdot 7) + 4 \cdot (5 \cdot 7) + 2 \cdot 7 + 1\right] \cdot 4$$

9. (15 points) Let `A[n][n]` be an upper triangular matrix in the example given below. The elements of this triangular matrix A are stored in the one-dimensional array U as shown.

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ 0 & a_{11} & a_{12} & a_{13} \\ 0 & 0 & a_{22} & a_{23} \\ 0 & 0 & 0 & a_{33} \end{bmatrix} \begin{matrix} \leftarrow \\ i \\ \} n-i-1 \end{matrix}$$

$U =$

a_{00}	a_{01}	a_{11}	a_{02}	a_{12}	a_{22}	a_{03}	a_{13}	a_{23}	a_{33}

(a) (~~3~~ 4 points) What is the number of items stored in U if A has n rows and n columns?

$$(N + (N-1) + \cdots + 1) = \frac{N \cdot (N+1)}{2} = \binom{N+1}{2}$$

(b) (~~4~~ 5 points) What is x if $U[x]$ stores the entry $A[i][j]$? Show your calculations.

Answer 1: $n + (n-1) + \cdots + (n-(i-1)) + (j-i) = n \cdot i - \frac{(i-1) \cdot i}{2} + j - i$

$= (n-1)i - \binom{i}{2} + j$

Answer 2: $\binom{n+1}{2} - \binom{n-i+1}{2} - (n-j)$

(c) (~~8~~ 6 points) Please fill in the blanks in the method `readtriangularmatrix(int[], int)` that reads integers from the keyboard and fills the one-dimensional array U with entries of A as shown in the example.

```
void readtriangularmatrix(int U[], int n)
{
     int i, j, k;
     if(n*(n+1)\ 2 > MAX_SIZE){
          printf("\n invalid array size \ n");
          exit(-1);
     }
     else
          for(i=0; i<=n-1; i++){
               k= .....
               for(j=0; j<=i; j++)
                    scanf("%d", ..... );
          }
}
```

→ $\binom{n+1}{2} - \binom{n-i+1}{2}$; (I)

→ &U[k-(n-j)] (II)

Or according to Answer 1:

(I): $(n-1)i - \binom{i}{2}$;

(II): &U[k+j]

10. (9 points) Assume a special type of matrix, named *stripe matrix*, is a matrix that has value 0 in all even-numbered columns (column 2, column 4, ...) as shown in the example below. (For simplicity, assume that all stripe matrices are N by N matrices and N is even.)

$$A = \begin{array}{c} \\ i=0 \\ i=1 \\ i=2 \\ i=3 \end{array} \begin{array}{c} \begin{array}{cccc} j=0 & j=1 & j=2 & j=3 \end{array} \\ \begin{bmatrix} 2 & 0 & 3 & 0 \\ 4 & 0 & 1 & 0 \\ 5 & 0 & -3 & 0 \\ 20 & 0 & 12 & 0 \end{bmatrix} \end{array}$$

(a) (5 points) Write a 1-dimensional array representation of the matrix above, by filling in the entries of the array U.

U =

U[0]	U[1]						U[7]		
2	4	5	20	3	1	-3	12	X	X

(b) (4 points) What is the value of x if $U[x]$ stores a_{ij}?

$$x = n \cdot \frac{j}{2} + i - 1$$

Check: a_{30} is at $U[3] = 20$ ✓

a_{32} is at $U[4+3] = U[7] = 12$ ✓

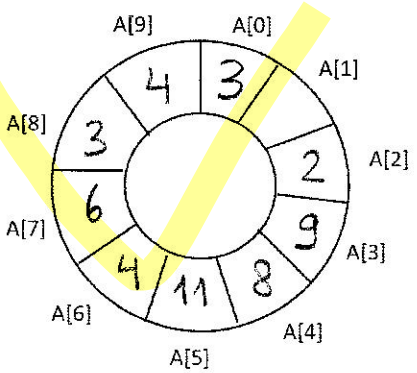
11. (8 points) The following operations are applied to an empty queue consecutively: `Enqueue(7)`, `Enqueue(5)`, `Enqueue(2)`, `Dequeue( )`, `Enqueue(9)`, `Enqueue(8)`, `Enqueue(11)`, `Enqueue(4)`, `Enqueue(6)`, `Enqueue(3)`, `Dequeue( )`, `Enqueue(4)`, `Enqueue(3)`

(a) (4 points) Given an array `int A[10]`, fill the entries of the array representing the queue obtained above. (Assume that the array is initially empty.) Can all operations be performed?

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
~~7~~	~~5~~	2	9	8	11	4	6	3	4

(We cannot insert 3.)

(b) (4 points) Given a *circular* array `int A[10]`, fill the entries of the circular array representing the queue obtained above. (Assume that the array is initially empty.) Can all operations be performed?



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);

}
```

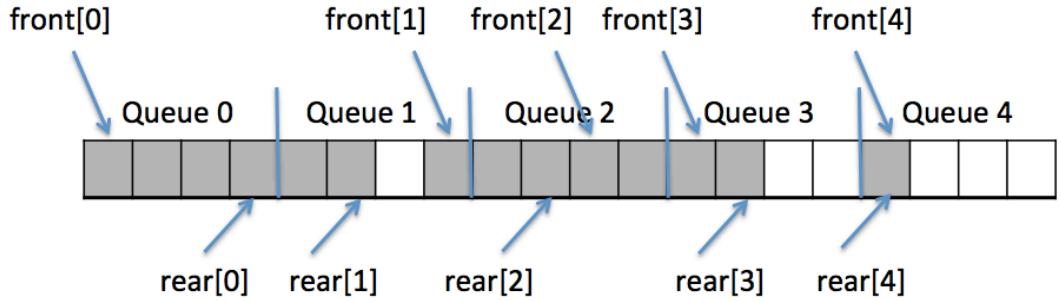
Question 3. Postfix is preferable than infix for computers because of having no checks for parenthesis and no check for operator precedence rules. Prefix has also the same properties as postfix for having no parenthesis and precedence check. Give a reason why postfix is more preferable than prefix for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

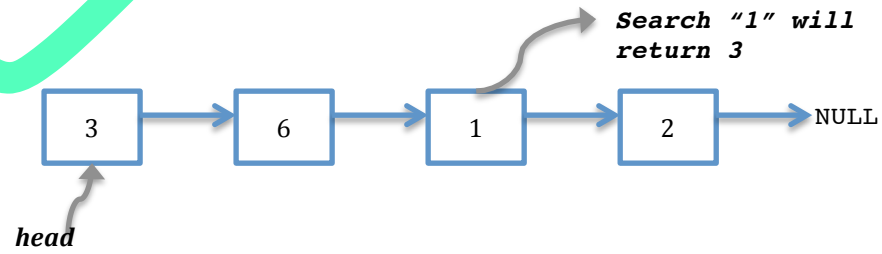
free_= 3
*list = 4

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)
}
```

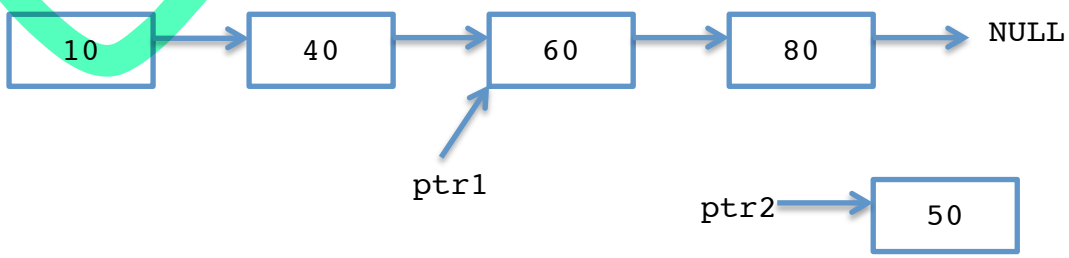
Question 10. Print the contents of the linked list returned by the Func function for the provided two linked lists and the code below.

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…

BBM201 – Data Structures – Fall 2015

Make-up Exam

28.12.2015 – 13:00

Name Surname: _____

Student ID : _____ **Section:**_____

Duration: 60 minutes

Question	1	2	3	4	5	Total
Points	24	12	14	30	20	100
Grade						

Question 1.

a) The following function calculates the number of r-combinations of n numbers, which is equal to the value of T[n,r] in the 2-dimensional array T. Find how many array accesses are made to T when the following function is executed.

```
function choose(n,r)
        for i:=0 to n-r do T[i,0]:=1;
        for i:=0 to r do T[i,i]:=1;
        for j:=1 to r do
                for i:=j+1 to n-r+j do
                        T[i,j]:=T[i-1,j-1]+T[i-1,j]
        return T[n,r]
```

b) The following code, called binsearch, searches a given number (**searchnum**) in a sorted list of **n** numbers. What is the worst-case, average-case and best-case running time of this algorithm in terms of n? Explain your answer.

```
int binsearch(int list[], int searchnum, int left, int right)
{
  int middle;
  while(left <= right){
    middle = (left + right)/2;
    switch(compare(list[middle], searchnum)){
      case -1 : left = middle + 1; break;
      case 0; return middle;
      case 1: rigt = middle –1;
    }
   }
  return -1;
}
#define COMPARE(x,y) (((x)<(y))? -1 ((x)==(y))?0:1)
```

Question 2. What is the output of the following code block if the stack consists of items {1,3,5,6,9}, where 1 is the top?

```
void enqueue(int queue[], int value);
int dequeue(int queue[]);
void push(int stack[], int value);
int pop(int stack[]);
bool isEmpty(int stack[]);



void methodA(int stack[5]){
  int queue[5],i;
  while(!isEmpty(stack))
     enqueue(queue, pop(stack));
  while(!isEmpty(queue))
     push(stack, dequeue(queue));
  while(!isEmpty(stack))
     printf("%d", pop(stack));
}
```

Question 3. Please convert the following prefix expression into infix and postfix notation:

/*-6/7+5/3+1*24-15-28

a) Infix notation:

b) Postfix notation:

Question 4. Fill the below Reverse function which reverses given singly linked list using recursion.

```
//global variables
struct Node * head;
    …

void main(){
        …
        Reverse(head);
        …
}

void Reverse (struct Node * p)
{









}
```

Question 5. According to the given array based linked list which is defined as follows:

```
#define MAX_LIST 10

typedef struct{
      char name[7];
      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
```

	name	link
[0]	“Banu”	1
[1]	“Irem”	4
[2]	“Zerrin”	-1
[3]	“Ali”	0
[4]	“Leyla”	6
[5]	“Ahmet"	3
[6]	“Mehmet”	2
[7]		8
[8]		9

[9]		-1

Imagine we begin with a sorted list given in the first table given below. After we insert “Adil” and delete “Leyla” 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 **Insert “Adil”** **Delete “Leyla”**

free_= 7 free_= free_=
*list = 5 *list = *list

	name	link
[0]		
[1]		
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		

	name	link
[0]		
[1]		
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		

BBM201 – Data Structures – Fall 2016
1st Midterm
11.11.2016

Name Surname: _____

Student ID :_____ **Section:**_____

Süre … dakikadır.

Question	1	2	3	4	5	6	7	8	Total
Points									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) A data structure is a way to store and organize data in computer, so that it can be used efficiently.

True ☐ False ☐

b) 3 n^2 + 10 n log n = O(n log n)

True ☐ False ☐

b) Big-Oh (O) notation gives a lower bound on the running time of a program.

True ☐ False ☐

c) The cost of insertion (push) and deletion (pop) of an element in a stack is O(1).

True ☐ False ☐

d) The complexity of the worst-case running time of binary search for an array of size n is n^2.

True ☐ False ☐

Question 2. A palindromic word is a sequence of characters that reads the same backward and forward. For example; *repaper*, *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 for stack operations, please fill in the gaps in the code.

```
void push(char);
void pop();

void IsPalindrome(char str[]){

      int len = strlen(str);
      for (i = 0; i < len; i++)
          _____________________________;

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

      if (_____________________)
          printf("%s is a palindromic string\n", str);
      else
          printf("%s is not a palindromic string\n", str);
}
```

Question 3.

Please write the output of the following method.

```
void recursiveFun(int value)
{
      if(0 < value && value < 10)
      {
              recursiveFun(value - 2);
              recursiveFun(value + 1);
              printf(" %d", value);
      }
}
```

Question 4.

Please write the output of the following method.

```
static int negative(int num)
{
    if(num >= 20)
        return -5;
    else
        return negative(num + 4) + 2 * num;
}
```

Question 5.
Let a[n][n] be an upper triangular matrix (see the example given below). The elements of this triangular matrix are stored in a one-dimensional array as given below:

a_{oo}	a_{o1}	a_{o2}	a_{o3}
0	a_{11}	a_{12}	a_{13}
0	0	a_{22}	a_{23}
0	0	0	a_{33}

U

a_{00}	a_{10}	a_{11}	a_{20}	a_{21}	a_{22}	a_{30}	a_{31}	a_{32}	a_{33}

Please complete the method readtriangularmatrix(int[], int) that reads integers from the keyboard and fills the one-dimensional array U with an upper triangular matrix.

```
void readtriangularmatrix(int U[], int n){

    int i, j, k;
    if(n*(n+1)/2 > MAX_SIZE){
       printf("\n invalid array size \n");
       exit(-1);
    }
    else
      for(i=0; i<=n-1; i++){
         k=……..
         for(j=0; j<=i; j++)
              scanf("%d", ……);
      }
}
```

Question 6.
If C is the array shown with its address above each node, write what the following lines of a program will print in the empty column.

100			116				132				
3	4	1	5	0	2	9	8	7	2	10	12

int C[3][2][2];

The code	will print:

printf("C=%d", C);	
printf("C+2=%d", C+2);	
printf("*(*C+1)) =%d", *(*C+1));	
printf(" *(C[0]+1)=%d ’’, *(C[0]+1)) ;	
printf(" *(C[2][1]+1)=%d ’’, *(C[2][1]+1));	
printf(" *(*((*(C+2))+1))=%d ’’, *(*((*(C+2))+1));	

BBM201 – Data Structures – Fall 2017
1st Midterm
02.11.2017 – 90 minutes

Name Surname: _____

Student ID : _____

Section: ☐ **Section 1 (Burcu Can)**

☐ **Section 2 (Sevil Şen)**

☐ **Section 3 (Adnan Özsoy)**

Questions	1	2	3	4	5	6	Total
Points	15	16	14	14	21	20	100
Grade							

1. (Recursion) ***(15 points)***

a. Write the output of the following code. ***(5 points)***

```
void mystery(int n){
        if(n==0) return;
        for(int i=0;i<n;i++){
             printf("%d", n);
             mystery(n-1);
       }
}
int main()
{
  mystery(2);
  return 0;
}
```

2 1 2 1

b. Please write the recursive method that applies exponentiation on a given integer x, so it returns x^m. Global variables are not allowed for the solution. ***(10 points)***

```
int exp(int x, int m) {

    if (m == 0)
         return 1;
     return x * exp(x, m-1);



}
```

2. (Stack/Queue) Given the initial empty position of stack and queue (circular) below, give the final representation of data below for array representations and fill the values of top, front and rear positions into an array of size 8. ***(16 points)***

a. Stack ***(8 points)***

Push (5) , Push (2) , Pop () , Push (6) , Push (3) , Pop () , Push (8) , Push (2) , Push (7) , Push (4) , Push (6) , Push (3) , Push (8)

0	1	2	3	4	5	6	7
5	6	8	2	7	4	6	3

	Top
Initial	-1
Final	7

b. Circular Queue ***(8 points)***

Enqueue(5) , Enqueue(2) , Dequeue () , Enqueue(6) , Enqueue(3) , Dequeue () , Enqueue(8) , Enqueue(2) , Enqueue(7) , Enqueue(4) , Enqueue(1)

0	1	2	3	4	5	6	7
1		6	3	8	2	7	4

	First	Rear
Initial	-1	-1
Final	2	0

3. (Pointers) If C is the array shown with its address above each node, write what the following lines of a program will print in the empty column. ***(14 points)***

			116						132		
3	4	1	5	0	2	9	8	7	2	10	12

int C[3][2][2];

The code	will print:
printf("C+1=%d", C+1);	**120**
printf("*(C+2)=%d", *(C+2));	**136**
printf("*(*(C+2)+1)) =%d", *(*(C+2)+1));	**144**
printf(" *(**C+1)=%d ’’, *(**C+1)) ;	**4**
printf(" *(C[2][1]+1)=%d ’’, *(C[2][1]+1));	**12**
printf(" *(*((*(C+1))+1))=%d ’’, *(*((*(C+1))+1));	**9**
printf("*(*(C+2)+2)) =%d", *(*(C+2)+2));	**Invalid**

4. (Performance) Please give time complexities of the functions given below, compare them in a decreasing order. ***(14 points)***

a. $5n^2 - 6n = O(n^2)$

b. $2n^2 + nlogn = O(n^2)$

c. $n^3 + 10^6n^2 = O(n^3)$

d. $n^k + n + n^klogn = O(n^klogn)$

e. O (n^3)

```
for (i=0; i<n; i++){
    for (j=0; j<=n-1; j++){
        for (k=j; k<=n-1; k++){
            … loop body…
        }
    }
}
```

f. O (nlogn)

```
for (i=0; i<n; i++){
    for (j=1; j<=n-1; j*2){
    … loop body …
    }
}
```

Order = n^klogn ($k>=3$) n^3 n^2 nlogn

5. (Problem Solving) Write a C function to find out the maximum and second maximum numbers from a given two-dimensional array of integers. Complete the code below where there are spaces "_____" as needed and inside the function TwoMax. Your answer should trace the array only once for full credit. ***(21 points)***

```
int main()
{
       int first=0, second=0;
       int **array;

       //allocate for the two-dimensional array
       array = malloc(rows * sizeof ( int *));
       for (int i=0; i<rows; i++)
        {
              x[i] = malloc(cols * sizeof(int));
        }
       initData(x);
       //assume this function fills the array x with random values

       TwoMax( ______ arr, rows, cols, & first, & second)

       printf("The maximum number is %d, second maximum is %d\n",
first, second);

       return 1;
}

// arr is two-dimensional array,
//n is the first dimension of the array,
//m is the second dimension of the array
int TwoMax( int ** arr, int n, int m, int * first, int * second)
{


  int i, j;
  *first = *second = arr[0][0];
  for(i = 0; i <n ;i++)
  {
    for(j = 0; j <m ; j++){

      if(arr[i][j] > *first)
      {
        *second = *first;
        *first = arr[i][j];
      }
      else if(arr[i][j] > *second && arr[i][j] < *first)
        *second = arr[i][j];
    }

  }




}
```

6. (Sparse Matrix) A matrix is called symmetric if for all values of i and j satisfies A[i][j] = A[j][i]. A sparse matrix has at least $\frac{m}{2} + 1$ zero values out of all m items in the matrix. Given that a matrix A is sparse, symmetric and square (NxN) ***(20 points)***

a. Propose and describe an efficient representation to improve the space complexity compared to two-dimensional representation of matrix A. ***(7 points)***

Use the same representation as sparse matrix, except show only lower triangle(or upper) non zero values

b. Justify and show mathematically your proposed representation is space efficient. ***(7 points)***

Since sparse has at most m/2 non zero, and we will only show half of the non zero values, soin maximum we will use m/4 values, using 3 pointers each, thus 3m/4 is better than m

c. Given the two-dimensional representation of A, describe and give pseudocode of converting two-dimensional representation into your proposed representation using minimum number of accesses to matrix A. Give the complexity of this operation in tilde (~) notation (Note: this part of the question looks for the most efficient approach for full credit) ***(6 points)***

Access as lower triangle only, no need to access the full matrix,
For i=0 to N, i++
For j=0 to j<=i, j++
Insert A[i][j] to representation
This will make Nx(N-1)/2 accesses thus ~N^2/2

BBM201 – Data Structures – Fall 2017
2nd Midterm
14.12.2017 – 13:00-15:00

Name Surname: _____

Student ID : _____ **Section:**_____

Duration: 120 minutes

Question	1	2	3	4	5	6	7	Total
Points	12	12	16	18	12	12	18	100
Grade								

Question 1. Evaluation of expressions.

a. If a= 10, b= 3, c=−2, evaluate the following postfix expression:

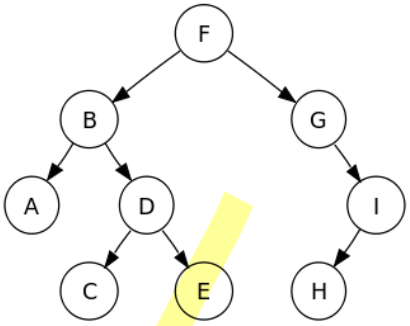
aa+bc-* = 100

b. Convert the following fully parenthesized infix expression to postfix and prefix notation:

(x-((x+y)*((z/r)+p)))

postfix : xxy+zr/p+*-
prefix : -x*+xy+/zrp

Question 2. Trees. Give the traversal order of the accessed elements for each traversal algorithm for the given tree.



Preorder: FBADCEGIH

Inorder: ABCDEFGHI

Postorder: ACEDBHIGF

Question 3. Array based linked list. According to the given array based linked list that is defined as follows:

```
#define MAX_LIST 10

typedef struct{
      char name[7];
      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 in the first table given below. After we insert "Adil" and delete "Leyla" successively in the same 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

	name	link
[0]	"Banu"	1
[1]	"Irem"	4
[2]	"Zerrin"	-1
[3]	"Ali"	0
[4]	"Leyla"	6
[5]	"Ahmet"	3
[6]	"Mehmet"	2
[7]		8
[8]		9
[9]		-1

free_= 7
*list = 5

Insert "Adil"

	name	link
[0]		
[1]		
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		

free_=
*list =

Delete "Leyla"

	name	link
[0]		
[1]		
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		
[8]		
[9]		

free_=
*list =

Question 4. ***Linked lists.*** The information of athletes applying for a long distance run will be stored as a linked list. Assume that the athelete list is not globally defined.

```
#define ARRAY_SIZE 21

typedef struct athlete {
      int no;
      char name[ARRAY_SIZE];
      double runningTime;
      struct athlete *next;
   } athleteType;
```

a. Write the function named ***printList*** that displays all the information of the athletes whose running times are smaller than the time given as a parameter to the function.

```
void printList(athleteType *list, double time)
{
      for(;list != NULL; list = list->next)
         if(list->runningTime < time)
            printf("%d  %s  %lf  \n",  list->no,  list->name,  list->runningTime);
}
```

b. Write the function named ***append*** that adds a new athlete with the given no, name and time information to the end of the list.

```
void append(athleteType **list, int no, char name[ARRAY_SIZE], double time)
{
    athleteType *temp = *list;
     athleteType                                          *newA=(athleteType*)malloc(sizeof(athleteType));

     newA->no = no;
     strcpy(newA->name, name);
     newA->runningTime = time;
     newA->next = NULL;


     if(*list == NULL)
        *list = new;
     else{
       for(;temp->next != NULL; temp = temp->next)
           ;
       temp->next = newA;
     }
}
```

c. Complete the main method given below according to the above implementations.

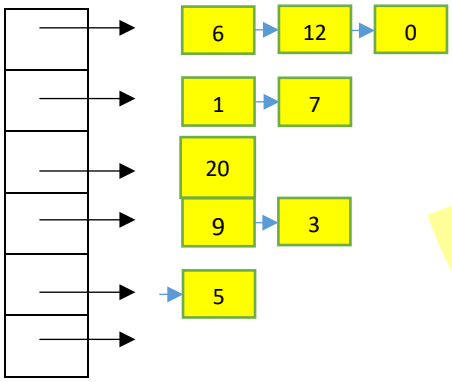
```
int main (void) {
   athleteType *list = NULL;
   append(  &list    , 10, "Ang", 14.80);//the list becomes 10 -> NULL
   append(  &list    , 8, "Appa", 16.20); //the list becomes 10 -> 8 -> NULL
   append(  &list   , 5, "Momo", 18.50);//the list becomes 10 -> 8 -> 5 -> NULL
   printList (   list , 17.00); //output : 10 Ang 14.80 , 8 Appa 16.20
    return 0;
}
```

Question 5. Hashtables.

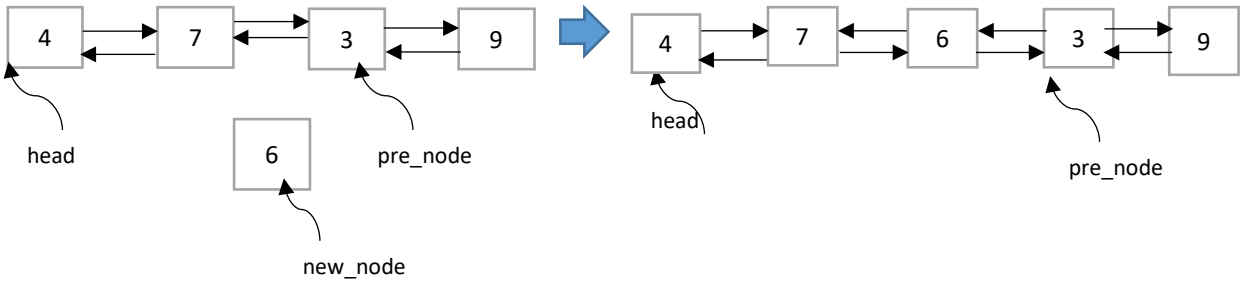
```
#define TABLE_SIZE 6

struct list *list_pointer;
typedef struct list {
    int key;
    list_pointer link;
};
list_pointer hash_table[TABLE_SIZE];

int hash_function(int key){
      return key % TABLE_SIZE;
}
void insert(int key)
{
      int hash_value = hash_function(key);
      list_pointer ptr, trail=NULL, lead=hash_table[hash_value];
      for (; lead; trail=lead, lead=lead->link)
            if (!strcmp(lead->key, key)) {
            printf("The key is in the table\n");
                  exit(1);
       }
       ptr = (list_pointer) malloc(sizeof(list));

        ptr->key = key;
        ptr->link = NULL;
        if (trail) trail->link = ptr;
        else hash_table[hash_value]= ptr;
}
```

According to the above hashtable implementation, fill in the below figure after the following method calls are applied:

insert(9); insert(20); insert(6);insert(12); insert(3); insert(5); insert(0); insert(1); insert(7);



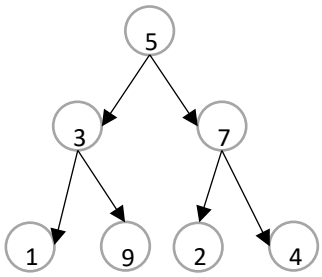
Question 6: Doubly linked list.



Write the method that inserts a new node before the given previous node in a doubly linked list. An example situation is given below.

```
struct node{
      int data;
      struct node* next;
      struct node* pre;
};
void insert(struct node* pre_node, struct node* new_node){

new_node->next = pre_node;
new_node->pre=pre_node->pre;
pre_node->pre->next = new_node;
pre_node->pre = new_node;

}
```

Question 7. Binary trees. We will use an array in order to store a full binary tree.



a) Fill in the given array that will include the tree contents given above.

5	3	7	1	9	2	4

b) Complete the given code that returns the nth node in the given level. For example, find_data(3, 2) will return 9, find_data(2, 1) will return 3, and so on.

```
#define ARRAY_SIZE 7
int tree[ARRAY_SIZE];
int pow (int x, int y); //assume the method is already defined.

int find_data(int level, int n){

return tree[pow(2, level-1)-1+n-1];


}
```

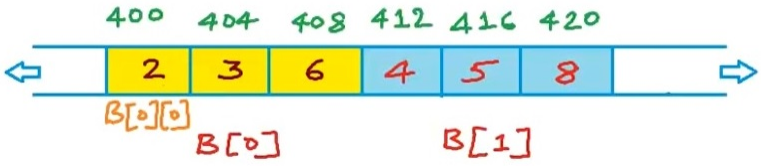
BBM 201
Exam 1
Time: 60 minutes

Answer the questions in the spaces provided on the question sheets. KEEP YOUR CELLPHONE TURNED OFF UNTIL THE EXAM IS OVER.

Name: _____

1. (15 points) If B is the array shown with its address above each node, write what the following lines of a program will print.

`int B[2][3]`:



   (a) (3 points) `printf`("%d", &B[1][1]);

   (b) (3 points) `printf`("%d", B+1);

   (c) (3 points) `printf`("%d", *(*B+1));

   (d) (3 points) `printf`("%d", B[1]+2);

   (e) (3 points) `printf`("%d", *(B+1)+1);

2. (12 points) Provide the time complexities below for the worst-case.

   (a) (4 points) Write the time complexity of inserting and deleting an element of an array.

   (b) (4 points) Write the time complexity of inserting and deleting an element of a linked-list.

   (c) (4 points) Write the time complexity of push(x), pop(), top() and IsEmpty() operations for an element x of a stack.

3. (9 points) How many bytes do the following data structures occupy in the memory?

   (a) (3 points) An INTEGER array of size 5.

   (b) (3 points) A CHARACTER doubly linked-list of size 5.

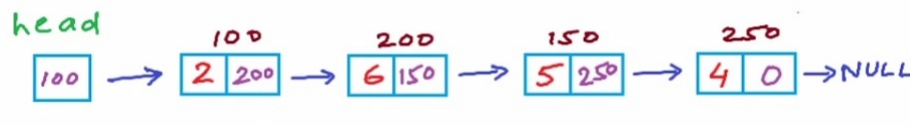
   (c) (3 points) A DOUBLE linked-list of size 5.

4. (12 points) (a) (6 points) Show how a stack looks after each of the following operations is applied consecutively. Write your answer in the stack column of the table below. Assume that this stack is initially empty.

   (b) (6 points) Write what `IsEmpty( )` and `Top( )` would return for the current stack if it was executed after each operation in the table below.

operation	current stack	IsEmpty( )	Top ( )
Push(2);			
Push(4);			
Pop( );			
Push(7);			
Push(3);			
Pop( );			

5. (16 points) (a) (12 points) Write each of the recursive calls made in their order when `Abc(head)` is executed in `main ()`. The address of each node is written above that node in this linked list.

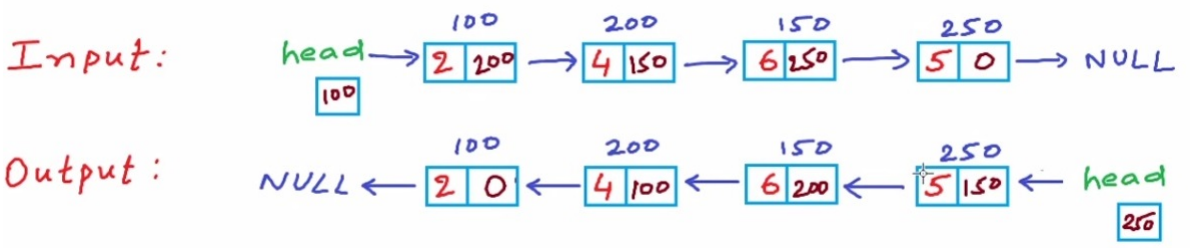



```
struct Node{
    int data;
    struct Node * next;
};
void Abc(struct Node * p)
{
    if(p == NULL)
    {
        return;
    }
    Abc(p → next);
    printf("%d", p → data);
}
int main()
{
    struct Node * head = NULL;
    head = Insert(head, 4);
    head = Insert(head, 5);
    head = Insert(head, 6);
    head = Insert(head, 2);
    Abc(head);
}
```

(b) (4 points) Write what `main()` will return in this program program.
(Note: `Insert(head, data)` inserts a new node in the beginning by putting `data` as the value of the head node.)

6. (36 points) The Reverse function below reverses any given linked list by using iteration method. The input argument of the Reverse function is the head of the linked list. In the following linked list, the number above each node indicates its address in the memory.

   (a) (20 points) If we execute the Reverse function for this linked list, redraw the changes on the input linked list after EACH ITERATION STEP of the while loop by showing the changes on the NODES and on the LINKS (arrows).




```
#include<stdio.h>
#include<stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* Reverse(struct Node* head) {
    struct Node *current,*prev,*next;
    current = head;
    prev = NULL;
    while(current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head = prev;
    return head;
}
```

   (b) (16 points) Write the value of addresses stored in "current", "prev" and "next" initially and after each iteration step of the while loop in the table below.

iteration	current	prev	next
1			
2			
3			
4			
5			

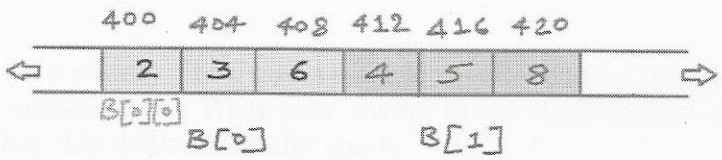
BBM 201
Exam 1
Time: 60 minutes

Answer the questions in the spaces provided on the question sheets. KEEP YOUR CELLPHONE TURNED OFF UNTIL THE EXAM IS OVER.

Name: SOLUTIONS

1. (15 points) If B is the array shown with its address above each node, write what the following lines of a program will print.

```
int B[2][3]
```



   (a) (3 points) `printf("%d", &B[1][1]);` 416

   (b) (3 points) `printf("%d", B+1);` 412

   (c) (3 points) `printf("%d", *(*B+1));` 3

   (d) (3 points) `printf("%d", B[1]+2);` 420

   (e) (3 points) `printf("%d", *(B+1)+1);` 416

2. (12 points) Provide the time complexities below for the worst-case.

   (a) (4 points) Write the time complexity of inserting and deleting an element of an array.

   Insert: O(n) Delete: O(n)

   (b) (4 points) Write the time complexity of inserting and deleting an element of a linked-list.

   Insert: O(n) Delete: O(n)

   (c) (4 points) Write the time complexity of push(x), pop(), top() and IsEmpty() operations for an element x of a stack.

   All are O(1)

3. (9 points) How many bytes do the following data structures occupy in the memory?

   (a) (3 points) An INTEGER array of size 5.

   (b) (3 points) A CHARACTER doubly linked-list of size 5.

   (c) (3 points) A DOUBLE linked-list of size 5.

a) $4 \times 5 = 20$

b) $(1 + 2 \times 4) \cdot 5 = 45$

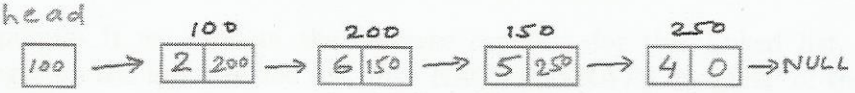
c) $(8 + 4) \cdot 5 = 60$

4. (12 points) (a) (6 points) Show how a stack looks after each of the following operations is applied consecutively. Write your answer in the stack column of the table below. Assume that this stack is initially empty.

   (b) (6 points) Write what `IsEmpty( )` and `Top( )` would return for the current stack if it was executed after each operation in the table below.

operation	current stack	IsEmpty( )	Top ( )
Push(2);	2	False	2
Push(4);	2,4	"	4
Pop( );	2	"	2
Push(7);	2,7	"	7
Push(3);	2,7,3	"	3
Pop( );	2,7	"	7

5. (16 points) (a) (12 points) Write each of the recursive calls made in their order when `Abc(head)` is executed in `main ()`. The address of each node is written above that node.




```
struct Node{
    int data;
    struct Node * next;
};
void Abc(struct Node * p)
{
    if(p == NULL)
    {
        return;
    }
    Abc(p -> next);
    printf("%d",p -> data);
}
int main()
{
    struct Node * head = NULL;
    head = Insert(head, 4);
    head = Insert(head, 5);
    head = Insert(head, 6);
    head = Insert(head, 2);
    Abc(head);
}
```

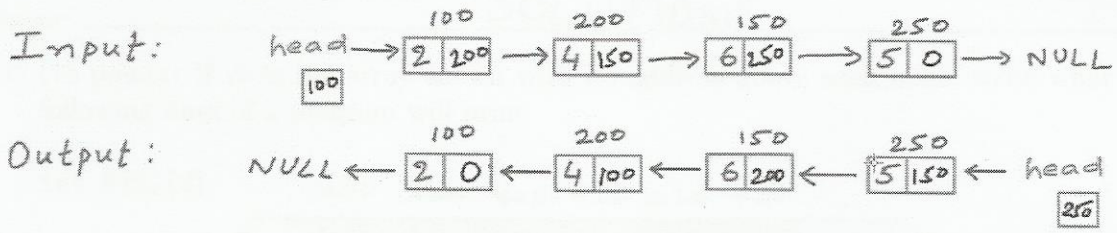
ABC(100)
↓
ABC(200)
↓
ABC(150)
↓
ABC(250)
↓
ABC(NULL)

(b) (4 points) Write what `main()` will return in this program program.
(Note: `Insert(head, data)` inserts a new node in the beginning by putting `data` as the value of the head node.)

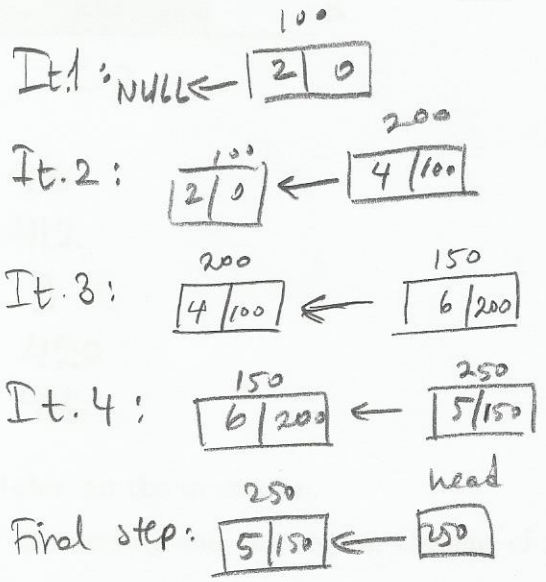
4 5 6 2

6. (36 points) The Reverse function below reverses any given linked list by using iteration method. The input argument of the Reverse function is the head of the linked list. In the following linked list, the number above each node indicates its address in the memory.

   (a) (20 points) If we execute the Reverse function for this linked list, redraw the changes on the input linked list after EACH ITERATION STEP of the while loop by showing the changes on the NODES and on the LINKS (arrows).




```
#include<stdio.h>
#include<stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* Reverse(struct Node* head) {
    struct Node *current,*prev,*next;
    current = head;
    prev = NULL;
    while(current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head = prev;
    return head;
}
```



   (b) (16 points) Write the value of addresses stored in "current", "prev" and "next" initially and after each iteration step of the while loop in the table below.

iteration	current	prev	next
1	100	NULL	X
2	200	100	200
3	150	200	150
4	250	150	250
5	NULL	250	NULL

BBM 201 – Data Structures - Fall 2019

1st Midterm

November 19, 2019

Name: _____

Student ID Number: _____ Section:_____

Problem	Points	Grade
1	15	
2	20	
3	15	
4	15	
5	20	
6	20	
Total	105	

INSTRUCTIONS

- Do not open this exam booklet until you are directed to do so. Read all the instructions first.
- When the exam begins, write your name on every page of this exam booklet.
- The exam contains six multi-part problems. You have **120 minutes** to earn 105 points (5pts bonus).
- The exam booklet contains **6 pages** including this one.
- This exam is an **open book and notes exam**. You are allowed to have two pieces of **bound** books or notebooks.
- Please write your answers in the space provided on the exam paper.
- Be neat.
- Good luck!

QUESTIONS

Question 1) Performance analysis (15 pts):

a) (3 pts) For a collection of algorithms that runs in O(1), O(n log n), O(n), $O(n^2)$, O(log n), O(n!), order the algorithms from fastest to slowest.

O(1), O(log n), O(n), O(n log n), $O(n^2)$, O(n!)

b) (12 pts) Find the big-O time complexity of each of the following code fragments.
Hint:

$$1 + 2 + 3 + \cdots + (n-2) + (n-1) = \frac{(n-1)(n)}{2} = O(n^2)$$

$$1 + 2 + 4 + \cdots + \frac{n}{4} + \frac{n}{2} + n = 2n - 1 = O(n)$$

```
int i = 1;
while (i <= n) {
      print("*");
      i = 2 * i;
}
```

Answer: O(log n)

```
int i = n;
while (i > 0) {
      for (int j = 0; j < n; j++)
            print ("*");
      i = i / 2;
}
```

Answer: O(n log n)

```
while (n > 0) {
      for (int j = 0; j < n; j++)
            print("*");
      n = n / 2;
}
```

Answer: O(n)

```
for (int i = 0; i < n; i++)
      for (int j = i+1; j > i; j--)
            for (int k = n; k > j; k--)
                  print("*");
```

Answer: O(n^2)

Question 2) Recursion (20 pts):

a) (12 pts) Write a recursive function (that takes a string and the length of the string as arguments) in C that returns 1 if the given string is palindrome, and 0 if it is not. A string is said to be a palindrome if the string read from left to right is equal to the string read from right to left. For example, "*kayak*", "*level*", "*radar*", "*repaper*", "*noon*" are all palindrome words in English, however "*paper*", "*book*", "*little*" are not. Only recursive solutions will be accepted.

```
int isPalindrome (char *str, int len){

    if (len == 0 || len == 1)
        return 1;
    if (str[0] == str[len-1])
        return isPalindrome (str+1, len-2);
    else return 0;
}
```

b) (8 pts) Please write the output of the following method if recursiveFun(6) is called.

```
void recursiveFun(int value)
{
      if(0 < value && value < 10)
      {
              recursiveFun(value - 2);
              recursiveFun(value + 1);
              printf(" %d", value);
      }
}
```

The code produces no output and ends with "segmentation fault".

Question 3) Multi-dimensional arrays (15 pts):

Consider the following multi-dimensional array:

int A[3][4][2] =

100 192

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Let p be defined as,

```
int (*p)[4][2] = A;
```

Fill in the values for the second column in the following table according to the expressions/code in the first column:

Expression/code	**Result**
`p`	100
`p + 2`	164
`*(p + 1)`	132
`**(p + 1)`	132
`***(p + 1)`	8
`*(*p + 2)`	116
`*(*(*p + 2) + 1)`	5
`*(*(*(p + 1) + 1) + 1)`	11

`int sum = 0;`
`for (int i = 0; i < 3; i++)`
`  sum += **(*(p + i) + 1);`
`printf("Sum : %d\n", sum);`
	Sum: 30

`int sum = 0;`
`for (int i = 0; i < 3; i++)`
`  sum += *(*(*(p + 2) + i) + 1);`
`printf("Sum : %d\n", sum);`
	Sum: 57

Question 4) Triangular Matrix (15 pts):

Let A[n][n] be a lower triangular matrix (see the example given below in A and C). The elements of this triangular matrix are stored in a one-dimensional array (U). We want to convert the given left-aligned lower triangular matrix into a right-aligned lower triangular matrix as given in B and D without changing the order of the items in the one-dimensional array U.

gettriangularmatrix(int i,int j,int n) method returns the item in the given index (A[i][j]).

Which lines in the method should we change to make it work for the right-aligned lower triangular matrix for the same one-dimensional array? Please write the updated lines on the side of each line that needs to be changed. No new lines will be added. Leave the space blank for the lines which do not require any changes.

A

a_{00}	0	0	0
a_{10}	a_{11}	0	0
a_{20}	a_{21}	a_{22}	0
a_{30}	a_{31}	a_{32}	a_{33}

B

0	0	0	a_{03}
0	0	a_{12}	a_{13}
0	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

C

1	0	0	0
2	3	0	0
4	5	6	0
7	8	9	10

D

0	0	0	1
0	0	2	3
0	4	5	6
7	8	9	10

U

a_{00}	a_{10}	a_{11}	a_{20}	a_{21}	a_{22}	a_{30}	a_{31}	a_{32}	a_{33}

U

a_{03}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{30}	a_{31}	a_{32}	a_{33}

```
int gettriangularmatrix(int i, int j, int n){

    if(i<0||i>=n||j<0||j>=n){  ..................................... 1
            printf("\n invalid index\n");........................... 2
            exit(-2);
     }
     else if(i>=j) //valid index      ........ else if (i+j>=n-1)..... 3
            return (i+1)*i/2+j  ........... (i*(i+1)/2)+j-(n-i-1)..... 4
     else return -1;          ..................................... 5
}
```

Question 5) Stacks (20 pts):

Answer the following questions by putting a check mark on the correct column. Unless otherwise stated, you have access to only the stack interface functions. You cannot directly access the underlying array structure.

Q	Question	True	False
1	To reverse the order of all the items within a stack, it is sufficient to use an additional stack (the items should end up in the original stack).		F
2	To process a list of tasks in the arriving order, using a single stack is sufficient.		F
3	Summing up all the items in a stack requires O(n) time.	T	
4	Finding the maximum item in a stack requires O(nlog n) time due to sorting.		F
5	Consider you have stacks S1 and S2 containing k1 and k2 integer values and both of them are sorted such that the item values increase towards the bottom of the stack. Let k be the sum of k1 and k2. Using only one more stack, S3, we want to merge both stacks, in same sorting order, in S1. We need at most 2*k **pop** operations to do this.	T	
6	To detect whether a word is a **palindrome** (which is same when read left-to-right and right-to-left) we can use a single stack. Some palindromes are racecar, radar, level, civic.	T	
7	When a stack of size n is full, we can transfer its contents to a larger stack in n pop and push operations.		F
8	For the 7^{th} question, if we had access to the underlying array structure, the stack capacity could be extended in constant O(1) time.		F
9	If we decide to replace a queue with stacks, we need two stacks to mimic the full queue functionality.	T	
10	Assume a **popBottom()** operation to pop the bottom item in a stack. If you have access to the underlying array structure, this operation can be implemented to require at most O(1) time.	T	

Question 6) Queues (20 pts):

a) (10 pts) A *priority queue* is a data structure that supports storing a set of values, each of which has an associated key. Each key-value pair is an entry in the priority queue. The basic operations on a priority queue are:

- *insert(k, v)* – insert value v with key k into the priority queue
- *removeMin()* – return and remove from the priority queue the entry with the smallest key

Other operations on the priority queue include *size()*, which returns the number of entries in the queue and *isEmpty()* which returns true if the queue is empty and false otherwise.

Two simple implementations of a priority queue are using an unsorted array, where new entries are added at the end of the array, and a sorted array, where entries in the array are sorted by their key values.

Fill in the following table to give the running times of the priority queue operations for these two implementations using *O()* notation. You should assume that the implementation is optimised.

Operation	Unsorted Array	Sorted Array
isEmpty()	O(1)	O(1)
insert(k, v)	O(1)	O(n)
removeMin()	O(n)	O(1)

b) (10 pts) For a given integer queue *Q*, fill in the blanks in the below pseudocode so that the function *findMaxOfQueue* will find the maximum element in *Q*. You may only use the operations: *enqueue(), dequeue(), size()*.

Usage:

dequeue() : Dequeues an item from Q

enqueue(X) : Enqueues X into Q

size() : Returns the size of Q

Queue must remain intact after finding the max. Each blank is either a complete single line or part of a single line.

```
findMaxOfQueue (Queue Q)
{
        max = ..... dequeue() ..... ;
        .....enqueue (max) ..... ;
        for ( int i=1 ; .....i<size()..... ; .....i++..... )
        {
                next = ..... dequeue().....;
                if (.....next>max.....) {
                        .....max=next..... ;
                }
                ..... enqueue(next)..... ;
        }
        return .....max..... ;
}
```

BBM 201 – Data Structures - Fall 2019

2nd Midterm

December 24, 2019

Name: _____

Student ID Number: _____ Section:_____

Problem	Points	Grade
1	20	
2	15	
3	20	
4	15	
5	19	
6	16	
Total	105	

INSTRUCTIONS

- Do not open this exam booklet until you are directed to do so. Read all the instructions first.
- When the exam begins, write your name on every page of this exam booklet.
- The exam contains six multi-part problems. You have **120 minutes** to earn 105 points (5pts bonus).
- The exam booklet contains **7 pages** including this one.
- This exam is an **open book and notes exam**. You are allowed to have two pieces of **bound** books or notebooks.
- Please write your answers in the space provided on the exam paper.
- Be neat.
- Good luck!

QUESTIONS

Question 1) Evaluation of Expressions (20 pts):

(10) a. Convert the following fully parenthesized infix expression to postfix and prefix notations:

(x-((x+y)^((z/w)-q)))

postfix:	**xxy+zw/q-^-**
prefix:	**-x^+xy-/zwq**

(10) b. Fill in the table below accordingly to convert the infix expression

(E-F)^A/C

to a postfix expression using an operator stack. Note that you are supposed to leave some cells blank (In other words, you should not fill all blank cells).

Token	Operator Stack			Output
	[0]	[1]	[2]	
(	(			
E	(			E
-	(	-		E
F	(	-		EF
)				EF-
^	^			EF-
A	^			EF-A
/	/			EF-A^
C	/			EF-A^C
eos				EF-A^C/

Question 2) Array-based Linked Lists (15 pts):

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

```
#define MAX_LIST 10

typedef struct{
        char name[10];
        int link;
      }planet;
planet 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 an alphabetically sorted list in the first table given below. In the sorted list, each element holds a link to the next element. That link points to the index of the next element. The first element in the list can be accessed through *list. The first available element in the list, where a new element can be inserted, is accessed through free_. The available items are also linked to each other in the same way.

After we insert "Jupiter" and delete "Neptune" successively (you will delete "Neptune" after you insert "Jupiter") in the same sorted list, what will be the contents of the list? Please fill in the tables and write down the new values of free_ and *list.

Initial Table

	name	link
[0]	"Venus"	-1
[1]	"Saturn"	0
[2]	"Neptune"	4
[3]	"Mars"	2
[4]	"Pluto"	1
[5]	"Earth"	3
[6]		7
[7]		8
[8]		9
[9]		-1

Insert "Jupiter"

	name	link
[0]	"Venus"	-1
[1]	"Saturn"	0
[2]	"Neptune"	4
[3]	"Mars"	2
[4]	"Pluto"	1
[5]	"Earth"	6
[6]	"Jupiter"	3
[7]		8
[8]		9
[9]		-1

Delete "Neptune"

	name	link
[0]	"Venus"	-1
[1]	"Saturn"	0
[2]	"Neptune"	7
[3]	"Mars"	4
[4]	"Pluto"	1
[5]	"Earth"	6
[6]	"Jupiter"	3
[7]		8
[8]		9
[9]		-1

free_ = 6 free_ = 7 free_ = 2

*list = 5 *list = 5 *list = 5

Question 3) Circular Linked Lists (20 pts):

(10) a. Complete the given method that returns 1 if the given linked list is circular, otherwise returns 0. Please note that an empty linked list is supposed to be circular. You are allowed to define only an extra variable without doing any memory allocation.

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

int isCircular(struct node* head){

if (head==null)
      return 1;

struct node* next = head->next;
while(node!=null && node !=head)
      node = node->next;
return (node == head);

}
```

(10) b. This time write a recursive method that returns 1 if the given linked list is circular, otherwise returns 0. Again an empty linked list is supposed to be circular. The parameter 'cur' points to the head node in the first call. You are not allowed to define any other variable.

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

int isCircular(struct node* head, struct node* cur){

if (head==null)
      return 1;

if(cur==null)
      return 0;

if(head==cur)
      return 1;

return isCircular(head, cur->next);

}
```

Question 4) Array of Linked Lists (15 pts):

For the program code given below, input of func is:

Hint: (int)log10(94634) = 4

(5) a. Write down the exact output line printed after **== PART 1 : ==**

2 >> 0 >> 3 >> 0 >> 1 >> 1 >> 0 >> 0 >> 0 >> 0 >>

(10) b. Write down the exact output line printed after **== PART 2 : ==**

>> 7 >> 2 >> 357 >> 235 >> 812 >> 45645 >> 545678

```
typedef struct ListNode* ListNode_pointer;
typedef struct ListNode {
    int element;
    ListNode_pointer next;
};

void func(ListNode_pointer *A) {
    if (*A == NULL) return;
    ListNode_pointer temp = *A;
    ListNode_pointer newHeads[10];
    ListNode_pointer newTails[10], tail;
    int co[10], om;
    for (int i=0; i<10; i++) {
        newHeads[i] = NULL;
        newTails[i] = NULL;
        co[i] = 0;
    }

    while (temp != NULL) {
        om = (int)log10(temp->element);

        co[om]++;
        if (newHeads[om] == NULL) {
            newHeads[om] = temp;
            newTails[om] = temp;
        }
        else {
            newTails[om]->next = temp;
            newTails[om] = temp;
        }
        temp = temp->next;
    }

    int i = 0;
    for( ; co[i] == 0; i++);
    *A = newHeads[i];
    tail = newTails[i];

    for (int j=i+1; j<10; j++) {
        if (newHeads[j] != NULL) {
            tail->next = newHeads[j];
            tail = newTails[j];
        }
    }
    tail->next = NULL;

    printf("\n== PART 1 : ==\n");
    for (int i=0; i<10; i++)
        printf("%d  >> ", co[i]);
    printf("\n************** \n");

    temp = *A;
    printf("\n== PART 2 : ==\n");
    for( ;temp != NULL; temp = temp->next)
        printf(" >> %d",temp->element);
    printf("\n************** \n");
}
```

Question 5) Linked List Implementation of Stack (19 pts):

Fill the following empty lines to implement a **recursive** function to find the minimum value stored in a linked list-based stack. You can assume that the stack is not empty. Each empty line is to be filled with one of the lines on the right. Only 6 lines from the right will be used. Write the corresponding line number inside the parenthesis.

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

int FindMin(struct Node* s) {
   ( 3 )
   ( 5 )
   ( 1 )
   (10 )
   {
      ( 6 )
      ( 2 )
   }
}
```

Each line is worth 1 point.
+3 points for 6 correct lines.
Alternatively, first 3 lines can be (5),(1),(3).

1	return s->data;
2	return s->data<min?s->data:min;
3	int min;
4	return min;
5	if (s->link == NULL)
6	min = FindMin(s->link);
7	if (s == NULL)
8	s = s->link;
9	min = FindMin(s);
10	else

Now implement the same **recursive** function assuming you have no access to the underlying linked list structure. You can only use the standard stack functions to access and modify a *global* stack. You can assume that the stack is not empty. Each empty line is to be filled with one of the lines on the right. Only 7 lines from the right will be used. Write the corresponding line number inside the parenthesis.

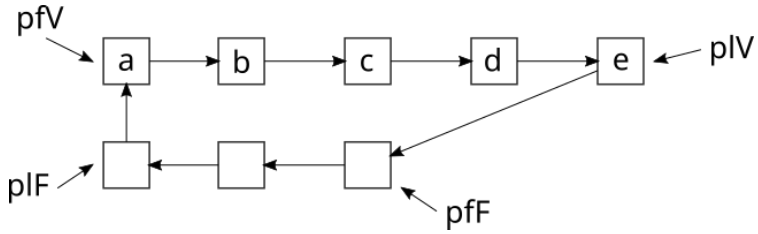
```
int FindMin()
{
   ( 9 )
   ( 1 )
   ( 7 )
   (10 )
   ( 4 )
   ( 2 )
   ( 8 )
}
```

Each line is worth 1 point.
+3 points for 7 correct lines.
Alternatively, (1) and (7) can be swapped.

1	pop();
2	push(a);
3	return minVal;
4	minVal = FindMin();
5	if(isEmpty())
6	push(minVal);
7	int minVal = a;
8	return a<minVal?a:minVal;
9	int a = top();
10	if(!isEmpty())

Question 6) Circular Linked List Implementation of Stacks and Queues (16 pts):

Consider a dynamically allocated linked list based circular data structure. As long as there are free nodes available, new node requests are answered by these nodes. The last node with a value is followed by the first free node and the last free node is followed by the first node with a value as shown in the figure below:



This structure can be implemented in many different ways based on the following possible pointers and/or values:

- store a pointer, pfV, to the first node with a value (top, front)
- store a pointer, pfF, to the first free node
- store a pointer, plV, to the last node with a value
- store a pointer, plF, to the last free node
- number of values stored in the list, nV
- number of free nodes, nF

Assuming that there are **m** nodes with values and **k** free nodes in the list (m>1, k>1), fill in the following table with the **number of nodes accessed or modified** (value or link fields are read or changed) during each ADT operation for the corresponding implementation variant, assuming a smart implementation. You can assume that the free nodes can be rearranged/reordered as necessary, but the nodes with values maintain their order at all times.

Stored pointers	enqueue	dequeue/pop	push	top/front
pfV & pfF	1	1	k	1
plV & plF	2	1	3	2
pfV & nV	m+1	1	m+k	1
plV & nF	2	0	k+1	k+2

Q1 Academic Honesty

0 Points

It is a violation of the Academic Integrity Code to look at any reference material other than your textbook and lecture notes, or to give inappropriate help to someone or to receive unauthorized aid by someone in person or electronically via messaging apps such as WhatsApp. Academic Integrity is expected of all students of Hacettepe University at all times, whether in the presence or absence of members of the faculty. Do NOT sign nor take this exam if you do not agree with the honor code.

Understanding this, I declare I shall not give, use or receive unauthorized aid in this examination.

Signature (Specify your name and surname as your signature)

Q2 Complexity

10 Points

Q2.1

2 Points

What is the complexity of the following code fragment in big-O notation?

```
for (int i = n; i > 0; i /= 2)
   count++;
```

○ $O(n)$

- ○ $O(n/2)$
- ◉ $O(\log_2 n)$
- ○ $O(n \log_2 n)$

Q2.2
2 Points

What is the big-O notation of the following complexity?

$n^3 + 10^6 n^2$

- ◉ $O(n^3)$
- ○ $O(n^2)$
- ○ $O(10^6 n^2)$
- ○ $O(10^6)$

Q2.3
2 Points

What is the complexity of the following code fragment in big-O notation?

```
for (i=0; i<n; i++){
    for (j=0; j<=n-1; j++){
        for (k=j; k<=n-1; k++){
            … loop body…
        }
    }
}
```

- ○ $O(n^2)$
- ○ $O(n^2 logn)$
- ◉ $O(n^3)$

- $O(n^2/2)$

Q2.4
2 Points

What is the big-O notation of the following complexity?

$n^k + n + n^k logn$

- $O(n^k)$
- **$O(n^k logn)$**
- $O(n^k + n)$
- $O(n^k + n^k logn)$

Q2.5
2 Points

What is the complexity of the following code fragment in big-O notation?

```
for (i=0; i<n; i++){
      for (j=1; j<=n-1; j*2){
      … loop body …
      }
   }
```

- $O(n^2)$
- $O(n^2 logn)$
- **$O(nlogn)$**
- $O(logn)$

Q3 Recursion

10 Points

Write the output of the following code.

```
int RecursiveFunc (int array[], int index, int n)
{
        int val1, val2;
        if ( n==1 )
                return array[index];
        val1 = RecursiveFunc (array, index, n/2);
        val2 = RecursiveFunc (array, index+(n/2), n-(n/2));
        if (val1 > val2)
                return val1;
        else
                return val2;
}

int main()
{
        int a[4] = {14,12,18,15,2,16,4,13};
        cout << RecursiveFunc(a,0,8) << endl;
}
```

18

Q4 Multidimensional Array

20 Points

A matrix is called symmetric if for all values of i and j satisfies $A[i][j] = A[j][i]$. A sparse matrix has at least m/2 + 1 zero values out of all m items in the matrix. Given that matrix A is sparse, symmetric, and square (NxN).

Q4.1

10 Points

Propose and describe an efficient representation to improve the space complexity compared to two-dimensional representation of matrix A.

Q4.2
10 Points

Justify and show mathematically your proposed representation is space efficient.

Q5 Struct/Stack/Tree
12 Points

Q5.1
2 Points

Can a Structure (struct) contain a pointer to itself?

- ○ Compilation error
- ○ Runtime error
- ◉ Yes
- ○ No

Q5.2
2 Points

The `sizeof` for a struct is always equal to the sum of `sizeof` of

The `sizeof` for a struct is always equal to the sum of `sizeof` of each individual member.

- ○ True
- ◉ False

Q5.3
4 Points

Suppose that an intermixed sequence of (stack) push and pop operations are performed. The push operations push the integers 0 through 9 in order; the pop operations print out the return value. Which of the following printed sequence could not occur?

- ○ 2 5 6 7 4 8 9 3 1 0
- ○ 4 3 2 1 0 5 6 7 8 9
- ◉ 4 6 8 7 5 3 2 9 0 1
- ○ 4 3 2 1 0 9 8 7 6 5
- ○ 2 1 4 3 6 5 8 7 9 0

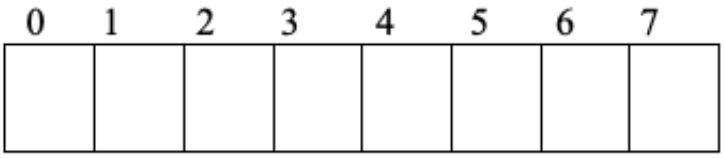
Q5.4
4 Points

What is the *maximum* height of a **full binary tree** with n nodes?

- ◉ $\lceil \log_2(n+1) \rceil$
- ○ $n/2$
- ○ $n-1$
- ○ n

Q6 Queue

10 Points



Enqueue(5) , Enqueue(2) , Dequeue () , Enqueue(6) , Enqueue(3) , Dequeue () , Enqueue(8) , Enqueue(2) , Enqueue(7) , Enqueue(4) , Enqueue(1)

Q6.1

5 Points

Given the initial empty position of the queue (circular) above, give the final representation of data below for array representations and fill the values of front and rear positions into an array of size 8.

Note: For your answer write each content of the array from left to right with spaces, and if there is a gap put - so if the array has empty places at index 3, 6, and 7 then a sample output will be: 1 2 3 - 4 5 - -

1 - 6 3 8 2 7 4

Q6.2

5 Points

	First	Rear
Initial	-1	-1
Final		

Write the values seperated by space e.g. 1 2

Write the values seperated by space e.g., 1 2

2 0

Q7 Data Structure

20 Points

You have a collection of documents and each document is composed of terms such as $D = (d_1, d_2, \ldots, d_{|D|})$ is the set of documents, $T_{d_i} = (t_1, t_2, \ldots, t_{|T_{d_i}|})$ is the set of terms within document d_i, and T is the set of unique terms in the collection. You would like to find the documents that contains the given set of terms $Q = \{(t^1, t^2, \ldots, t^k) | t^i \in T, k < |T|\}$.

Example:
$D = (d_1, d_2, d_3)$ where
T_{d_1} =(computer, engineer, data, structures, analysis)
T_{d_2} =(computer, science, database)
T_{d_3} =(computer, science, AI, data, algorithms)
T =(AI, algorithms, analysis, computer, data, database, engineer, science, structures)
The query Q =(computer, data) returns (d_1, d_3).

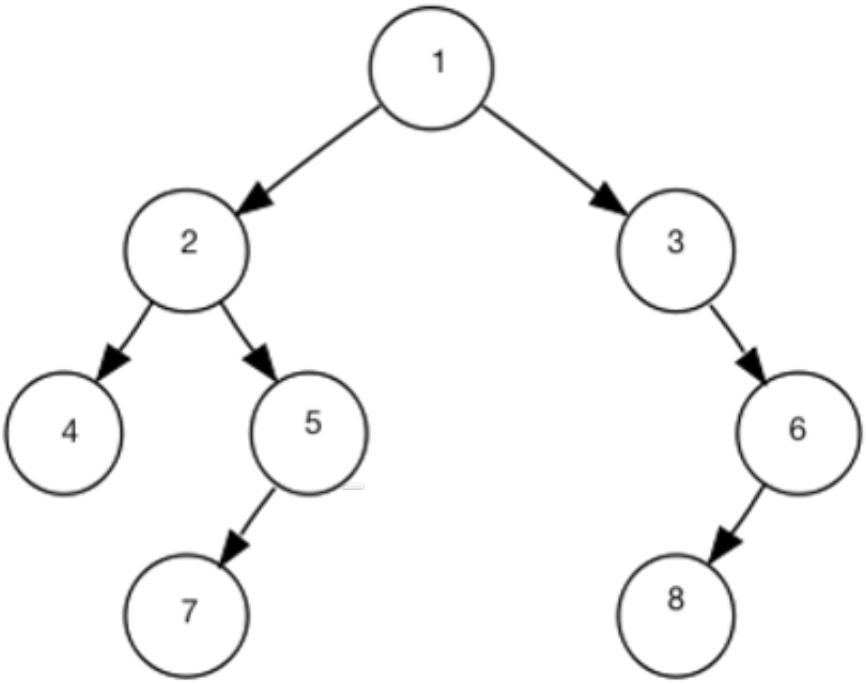
Choose a data structure to use in order to perform the query efficiently if $|T_{d_i}| \ll |T|, 1 \leq i \leq |D|$? Explain your reasoning in detail.

The space complexity of your data structure should be less than $O(|D||T|)$, otherwise you will not get any credit.
Note that $|.|$ denotes cardinality operator i.e., the number of elements.

Q8 Binary Tree

18 Points

For each of the below traversal algorithms, give the order of the nodes that are visited for the below tree structure.
Write the solution as a space-separated string e.g., 1 2 3 4 5 6 7 8



Q8.1

6 Points

Preorder:

1 2 4 5 7 3 6 8

Q8.2

6 Points

Inorder:

4 2 7 5 1 3 8 6

Q8.3
6 Points

Postorder:

4 7 5 2 8 6 3 1

Midterm Exam

● GRADED

STUDENT

TOTAL POINTS
77 / 100 pts

QUESTION 1
Academic Honesty **0** / 0 pts

QUESTION 2
Complexity **10** / 10 pts

2.1 (no title) **2** / 2 pts

2.2 (no title) **2** / 2 pts

2.3 (no title) **2** / 2 pts

2.4 (no title) **2** / 2 pts

2.5 (no title) **2** / 2 pts

QUESTION 3
Recursion **10** / 10 pts

QUESTION 4

Multidimensional Array **17** / 20 pts

4.1 (no title) **7** / 10 pts

4.2 (no title) **10** / 10 pts

QUESTION 5

Struct/Stack/Tree **12** / 12 pts

5.1 (no title) **2** / 2 pts

5.2 (no title) **2** / 2 pts

5.3 (no title) **4** / 4 pts

5.4 (no title) **4** / 4 pts

QUESTION 6

Queue **10** / 10 pts

6.1 (no title) **5** / 5 pts

6.2 (no title) **5** / 5 pts

QUESTION 7

Data Structure **0** / 20 pts

QUESTION 8

Binary Tree **18** / 18 pts

8.1 (no title) **6** / 6 pts

8.2 (no title) **6** / 6 pts

8.3 (no title) **6** / 6 pts

Q1
0 Points

It is a violation of the Academic Integrity Code to look at any reference material other than your textbook and lecture notes, or to give inappropriate help to someone or to receive unauthorized aid by someone in person or electronically via messaging apps such as WhatsApp. Academic Integrity is expected of all students of Hacettepe University at all times, whether in the presence or absence of members of the faculty. Do NOT sign nor take this exam if you do not agree with the honor code.

Understanding this, I declare I shall not give, use or receive unauthorized aid in this examination.

Signature (Specify your name and surname as your signature)

While answering the following questions, please consider the implementations that we discussed in our lectures unless stated otherwise.

Q2
8 Points

Q2.1
2 Points

Which of the following statements are true?

- [ ] Recursive functions run faster than non-recursive functions.
- [x] Recursive functions usually takes more memory space than non-recursive functions.
- [x] A recursive function can always be replaced by a non-recursive function.
- [x] In some cases, however, using recursion enables you to give a natural, straightforward, simple solution to a program that would otherwise be difficult to solve.

Q2.2

2 Points

Fill in the code to complete the following function for checking whether a string is a palindrome.

- ○ isPalindrome(s)
- ○ isPalindrome(s, low, high)
- ○ isPalindrome(s, low + 1, high)
- ○ isPalindrome(s, low, high - 1)
- ◉ isPalindrome(s, low + 1, high - 1)

Q2.3

2 Points

What is the output of the following program?

- ○ 3 33
- ○ 11
- ○ 5 6
- ◉ 5 33
- ○ 33 5

Q2.4
2 Points

If a set of the same elements is inserted into a binary search tree in two different orders, which of the following statements are true?

- ○ The two corresponding binary trees look the same.
- ◉ The inorder traversal generate the same sequence of nodes.
- ○ The preorder traversal generate the same sequence of nodes.
- ○ The postorder traversal generate the same sequence of nodes.

Q3
10 Points

Propose a data structure that supports the stack push and pop operations and a third operation findMin, which returns the smallest element in the data structure, all in O(1) worst-case time.

Ordered lists. Pop operation returns the first element in the data when removes the first element. The first element of ordered list is the minimum element of ordered list.

Q4
20 Points

Write a C++ function to return the array of integer data elements of a pointer based binary tree in level-order. Firstly the root, then nodes at level 2, followed by nodes at level 3, and so on. Your solution should work in linear time.

```
// tree = root of tree, tree->left returns left child and tree->right return right child.
// I try to convert this tree to array based tree. Thus we can see all elements in level order.
// Some positions in array can be null. (If left node is null or right node is null)

#define MaxSize 100
int arr[MaxSize];
arr[0] =  tree->item;

void tree_to_array(TreeNode *tree, int level)
{
    if (tree->left != nullptr)
    {
        arr[level * 2 + 1] = tree->left->item;
    }

    if (tree->right != nullptr)
    {
        arr[level * 2 + 2] = tree->right->item;
    }

    tree_to_array(tree->left, level * 2 + 1);
     tree_to_array(tree->right,level * 2 + 2);
}
```

Q5
9 Points

True/False

Q5.1
1 Point

The cost of insertion (Push) and deletion (Pop) of an element in a stack is O(1).

- [x] True
- [ ] False

Q5.2
1 Point

The complexity of the worst-case running time of binary search for an array of size n is n

- [ ] True
- [x] False

Q5.3
1 Point

$3n^2 + 10nlogn = O(nlogn)$.

- [ ] True
- [x] False

Q5.4
1 Point

Reversing a string is an application of stack.

- [x] True
- [ ] False

Q5.5
1 Point

A queue behaves on the basis of LIFO principle.

○ True

◉ False

Q5.6

1 Point

Items come out of a stack in the same order they go in.

○ True

◉ False

Q5.7

1 Point

The top operation does not modify the contents of a stack.

◉ True

○ False

Q5.8

1 Point

An expression has balanced parentheses if it contains an equal number of opening and closing parentheses.

○ True

◉ False

Q5.9

1 Point

When a function calls another function then the details of previous function are stored in Stack ?

- [x] True
- [ ] False

Q6
16 Points

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

Imagine we begin with a sorted list given in the first table below.

Representation format:
letter = D E F A B G I K _ _
link = 1 2 5 4 0 6 7 -1 9 -1

Q6.1
3 Points

After we insert “C” in the correct place for sorted order, what will be the contents of the list?
Please write the contents (letter and link) of the table as shown above ***representation format*** and write down the values of free_ and *list. (Give your answers in 6.1, 6.2, 6.3, and 6.4)

Note: Do not forget to use representation format:

letter =

D E F A B G I K C _

Q6.2
3 Points

link =

1 2 5 4 8 6 7 -1 0 -1

Q6.3
1 Point

free =

9

Q6.4
1 Point

*list =

3

Q6.5
3 Points

After all above operations, we delete “A” and keep the sorted order, what will be the contents of the list? Please write the contents (letter and link) of the table in the above representation format and write down the values of free_ and *list.(Give your answers in 6.5, 6.6, 6.7 and 6.8)

Note: Do not forget to use representation format:

letter =

D E F _ B G I K C _

Q6.6
3 Points

link =

1 2 5 -1 8 6 7 -1 0 -1

Q6.7
1 Point

free =

3

Q6.8
1 Point

*list =

4

Q7
6 Points

Complexity

Q7.1
3 Points

How many array accesses do the following code fragments make as a function of N?

- ~$(logn)$
- ~$(n \log n)$
- ~(n^2)
- ~$(n^2/2)$
- ~$(2n^2)$

Q7.2
3 Points

How many array accesses do the following code fragments make as a function of N?

- ○ $\sim(logn)$
- ◉ $\sim(2n \log n)$
- ○ $\sim(n \log n)$
- ○ $\sim(n^2/2)$
- ○ $\sim(n^2)$

Q8
5 Points

Given the following pseudocode for creating and inserting into a Binary Search Tree (BST):

```
// makenode: creates a node with the given integer data and returns the
tree = makenode(8);
// insertBST: inserts into BST considering the BST property.
insertBST(tree, 10);
insertBST(tree, 4);
insertBST(tree, 9);
insertBST(tree, 6);
insertBST(tree, 2);
insertBST(tree, 15);
insertBST(tree, 3);
insertBST(tree, 14);
insertBST(tree, 5);
```

Assuming that the BST is created properly using the pseudocode above, answer the following questions:

*(Note that, when we mention **node x** in the upcoming questions, it refers to **the node that contains x in its data section**.)*

Q8.1
1 Point

Write the data value of the left child of the node **10**.

9

Q8.2
1 Point

Write the data value of the right child of the node **4**.

6

Q8.3
1 Point

Is this tree balanced?

- (•) True
- ( ) False

Q8.4
1 Point

What is the height of this tree?

4

Q8.5
1 Point

Write the data value of the **first ancestor** of the node **15**.

10

Q9
6 Points

Assume that we represent a queue as a linked list. Write the pseudocode of the insert operation (`insert(queue, x)`) using `getnode()`, `info(p)`, `next(p)` pseudo functions that we used to represent linked lists.
*(Remember; **getnode()**: creates an empty node, **info(p)**: the data*

section of node p, ***next(p)****: the address of the next list item of node p. Assume that queue has* ***rear*** *and* ***front*** *members.)*

```
insert(queue, x)
{
   Node n = getnode();
   n->data = x;
   n->next = nullptr;
   rear->next = n;
   rear = n;
}
```

Q10

20 Points

Suppose we have **three binary operators**, namely **{&, ?, !}**, that work with integer operands, which are represented with capital letters for simplicity here (***e.g. A, B, C, .. so on***). Assume that the precedence of these operators as defined as follows: **& > ? > !**

Also assume that **all the operators are associated from left to right**: For instance, A & B & C is processed as ((A & B) & C).

You are expected to provide a pseudocode of an expression evaluation algorithm **using stacks**; ***considering the given precedence order and the association properties of the operators***. The algorithm is expected to process an expression string, which contains a sequence of valid symbols from a given expression, to return the correct value of the expression. (**Hint**: Use two stacks in your algorithm; one for the operators and one for the operands, to encode precedence of the operators properly.)

In your answer; **first start by a brief explanation of your algorithm**, then provide the **pseudocode** of your solution.

Supplementary Information:

A sample valid expression string: `A ? B & C & D ? E ! F ! G ? H`
The expected evaluation order is depicted using parenthesis:
`((((A ? ((B & C) & D)) ? E ) ! F) ! (G ? H))`

You can use the following helper functions in your pseudocode (if needed):

`Eval(operator, operand1, operand2)`: returns the true value of the given expression in the form of operand1 operator operand 2, **e.g.** Eval(&, B, C) is equivalent to the value of the expression B & C.

`Precedence(operator)`: returns a constant integer considering the precedence of operators, **e.g.** Precedence(&) > Precedence(?) is True.

`Next-token(expr_string)`: returns the current token in the input expression string, i.e. an operand or an operator. The position of the current token is internally kept track of in this function; **e.g.** for the expr_string “B ? C”; Next-token(expr-string) returns B first; then, Next-token(expr-string) returns ?, and so on. When there is no more tokens left, it returns **null**.)

Also, **assume that all stack functions are available for use**.

(Write your assumptions regarding the details of the functions provided above (if needed) and define all other helper functions that you use in your pseudocode (if needed).)

This program should run as stack but we should add some methods
(remove from any position) to this structure. Firstly, we store all elements and operations, then we operate all operations in the precedence order.

// stack "names" is the letter stack (A B C...), and stack "operations" (? & &...)is the operation stack.

// head1: first element of stack names head2: first element of stack operations
// delete method deletes element and return it.

```
name = head1; operation = head2;
while
if (precedence(operations->operation) >
precedence(operations->next->operation)
   eval(pop(operation), delete(), delete ());
   insert( (eval(operation, delete(), delete ())), name->next;
   name = head1; operation(head2);
name = name->next; operation = operation ->next;
```

Midterm Exam

GRADED

STUDENT

TOTAL POINTS

45 / 100 pts

QUESTION 1

(no title) **0** / 0 pts

QUESTION 2

(no title) **8** / 8 pts

2.1 (no title) **2** / 2 pts

2.2 (no title) **2** / 2 pts

2.3 (no title) **2** / 2 pts

2.4 (no title) **2** / 2 pts

QUESTION 3

(no title) **0** / 10 pts

QUESTION 4

(no title) **0** / 20 pts

QUESTION 5

(no title) **9** / 9 pts

5.1 (no title) **1** / 1 pt

5.2 (no title) **1** / 1 pt

5.3 (no title) **1** / 1 pt

5.4 (no title) **1** / 1 pt

5.5 (no title) **1** / 1 pt

5.6	(no title)	**1** / 1 pt
5.7	(no title)	**1** / 1 pt
5.8	(no title)	**1** / 1 pt
5.9	(no title)	**1** / 1 pt

QUESTION 6

(no title) **13** / 16 pts

6.1	(no title)	**3** / 3 pts
6.2	(no title)	**3** / 3 pts
6.3	(no title)	**1** / 1 pt
6.4	(no title)	**1** / 1 pt
6.5	(no title)	**3** / 3 pts
6.6	(no title)	**0** / 3 pts
6.7	(no title)	**1** / 1 pt
6.8	(no title)	**1** / 1 pt

QUESTION 7

(no title) **6** / 6 pts

7.1	(no title)	**3** / 3 pts
7.2	(no title)	**3** / 3 pts

QUESTION 8

(no title) **5** / 5 pts

8.1	(no title)	**1** / 1 pt
8.2	(no title)	**1** / 1 pt
8.3	(no title)	**1** / 1 pt
8.4	(no title)	**1** / 1 pt
8.5	(no title)	**1** / 1 pt

QUESTION 9

(no title) **4** / 6 pts

QUESTION 10

(no title) **0** / 20 pts

BBM201-MIDTERM EXAM 1
17 November 2021
SOLUTIONS

Q3. Propose a data structure that supports the stack push and pop operations and a third operation findMin, which returns the smallest element in the data structure, all in O(1) worst-case time.

Solution:

Let E be our extended stack. We will implement E with two stacks. One stack, which we'll call S, is used to keep track of the push and pop operations, and the otherM, keeps track of the minimum. To implement E.push (x), we perform S.push (x). If x is smaller than or equal to the top element in stack M, then we also perform M.push (x). To implement E.pop( ) we perform S.pop( ). If x is equal to the top element in stack M, then we also M.pop( ). E.findMin( ) is performed by examining the top of M. All these operations are clearly O(1).

Q4. Write a C++ function to return the array of integer data elements of a pointer based binary tree in level-order. Firstly the root, then nodes at level 2, followed by nodes at level 3, and so on. Your solution should work in linear time.

Solution:

```
int* printLevelOrder(Node* root)
{
    if (root == NULL)
        return;

    vector<int> vect;
    Queue<Node*> q;

    q.enqueue(root);

    while (!q.isEmpty()) {
        Node* node = q.dequeue();
        vect.push_back(node);

        if (node->left)
            q.enqueue(node->left);

        if (node->right)
            q.enqueue(node->right);
    }

// or alternatively return vect.data()
    int result[]  = new int[vect.size()];
    for(int i = 0; i < vect.size(); i++){
        result[i] = vect[i]
    }

    return result;
}
```

Q9-Assume that we represent a queue as a linked list. Write the pseudocode of the insert operation (insert(queue, x)) using getnode(), info(p), next(p) pseudo functions that we used to represent linked lists. *(Remember; **getnode()**: creates an empty node, **info(p)**: the data section of node p, **next(p)**: the address of the next list item of node p. Assume that the queue has **rear** and **front** members.)*

Solution:

```
insert(queue, x)
{
        p = getnode();
        info(p) = x;
        next(p) = null;
        if(queue.rear == null)
                queue.front = p;
        else
                next(queue.rear) = p;
        queue.rear = p;
}
```

Q10-Suppose we have **three binary operators**, namely **{&, ?, !}** that work with integer operands. Let us represent the integer operands,which are represented with capital letters for simplicity here (***e.g. A, B, C, .. so on***). Assume that the precedence of these operators are defined as follows: **& > ? > !**

Also assume that **all three operators are associated from left to right:** For instance, A & B & C is processed as ((A & B) & C).

You are expected to provide a pseudocode of an expression evaluation algorithm **using stacks**; ***considering the given precedence order and the association properties of the operators***. The algorithm is expected to process an expression string, which contains a sequence of valid symbols from a given expression, to return the correct value of the expression. (**Hint**: Use two stacks in your algorithm; one for the operators and one for the operands, to encode the precedence of the operators properly.)

In your answer; **first start by a brief explanation of your algorithm**, then provide the **pseudocode** of your solution.

Supplementary Information:

A sample valid expression string: A ? B & C & D ? E ! F ! G ? H
The expected evaluation order is depicted using parenthesis: ((((A ? ((B & C) & D)) ? E) ! F) ! (G ? H))

You can use the following helper functions in your pseudocode (if needed):
Eval(operator, operand1, operand2): returns the true value of the given expression in the form of operand1 operator, operand 2, e.g. Eval(&, B, C) is equivalent to the value of the expression B & C.
Precedence(operator): returns a constant integer considering the precedence of operators, **e.g.** Precedence(&) > Precedence(?) is True.
Next-token(expr_string): returns the current token in the input expression string, i.e. an operand or an operator. The position of the current token is internally kept track of in this function;
e.g. for the expr_string “B ? C”; Next-token(expr-string) returns B; then, Next-token(expr-string) returns ?, and so on. When there are no more tokens left, it returns **null**.)

Also, assume that all stack functions are available for use.
(Write your assumptions regarding the details of the functions provided above (if needed) and define all other helper functions that you use in your pseudocode (if needed).)

Solution:
// Assuming we have two stacks;
// operandS : operand stack
// operatorS : operator stack
/ Alg: While iterating over the tokens of the expr-string, each operand is pushed into operandS, each operator is pushed in operatorS if the current operator has higher precedence than the one on the stack top. Otherwise (i.e. less than or equal to the stack top), first evaluate the expression for the operator at the stack top, then push the current operator into the stack. (Expr. evaluations require popping two operands from the operandS and popping the operator at the stack top and pushing the expr value to the operandS) */*

```
ExpressionEvaluationAlg(expr-string)
{
        while(next=Next-token(expr-string)){

           if(is_operand(next))
                operandS.push(next)
            else{

                if (empty(operatorS))
                        operatorS.push(next);
                else{

                        top_opt = top(operatorS);
                        if (Precedence(next)>Precedence(top_opt))
                                operatorS.push(next);
                        else{
                                // evaluate the expression
                                opt = operatorS.pop();
                                op2 = operandS.pop();
                                op1 =  operandS.pop();
                                res = Eval(opt, op1, op2);
                                operarandS.push(res);
                                operatorS.push(next);
                         }// else
                }// else
         }// else

    }// while
    // empty the operator stack (in case there is any remaining operators inside)
    while(!operatorS.empty()){
                opt = operatorS.pop();
                op2 = operandS.pop();
                op1 =  operandS.pop();
                res = Eval(opt, op1, op2);
                operarandS.push(res);
        }

        val = operandS.pop();

    return val;
}
```

Easy

1. Name 5 stages of a program life cycle?
2. f(n)=4n3+6n2+7n , what is the algorithmic complexity of f(n) in O (Big Oh) notation
3. What is the memory address of y[2][2][3] if the memory address of y[0][0][0] α and y is declared as y[5][4][4]?
4. Trace the given recursive call as we did in class

```
int sum( int arr[], int n )
{
        if ( n == 0 )
                return 0;
        else
        {
                int smallResult = sum( arr + 1, n - 1 ); // "A"
                return smallResult + arr[ 0 ];
        }
}
```

Medium

1. Trace the given recursive call as we did in class and find what RecursiveFunc do in general?

```
int RecursiveFunc (int array[], int index, int n)
{
        int val1, val2;
        if ( n==1 )
                return array[index];
        val1 = RecursiveFunc (array, index, n/2);
        val2 = RecursiveFunc (array, index+(n/2), n-(n/2));
        if (val1 > val2)
                return val1;
        else
                return val2;
}

.. . .
int a[7] = {1,2,10,15,16,4,8};
printf( " value is %d\n", RecursiveFunc(a,0,7));
```

Solution: 16 - function finds the maximum

2. Write a recursive version of the below iterative solution of Fibonacci sequence code? Compare iterative and recursive version. Give two advantages for both.

```
int Fib(int n)
{

   int prev1, prev2, tem, j;

   if(n==0 || n==1)
      return n;
   else{
       prev1=0;
       prev2=1;
       for(j=1;j<=n;j++)
       {
           temp = prev1+prev2;
           prev1=prev2;
           prev2=temp;
       }
       return prev2;
   }
}
```

3. What is the order of growth of the worst-case running time of each of operation below?

2-Sum	
3-Sum	
Iterative Fibonacci	
Binary Search	

Solution: N^2 , n^3, n, logn

4. Approximately how many array accesses as a function of input size N?
Give some code here similar to slides

5. Given an array A[2][3] and the main memory represented as M[],
   In row major ordering what is the index of A[1][2] in M?
   In column major ordering what is the index of A[1][2] in M?

6. Write the output of below code segment? Assume the address of z[0][0] is 100.

```
int z[4][2] = { {2,4}, {6,8}, {1,3}, {5,7}};
printf("z[0] = %p\n", z[0]);
printf("  *z = %p\n", *z);
printf("   z = %p\n", z);
printf("   z[2][1] = %d\n", z[2][1]);
```

7. For the same above question, what is the memory address of z + 1 ?

8. A(x) and B(x) are two polynomials defined as
   A(x) = 3x ^20 + 2x^ 3 + 4 B(x) = x^5 +10x ^3 +3x ^2 +1
   In class we have shown a better structure to store polynomials.
   Ask to compare the analysis of addition operation and storage for new and old structure.

9. Assume a new type of special matrix, named stripe matrix, that has non-zero values in odd numbered columns as shown in below.
   a) Design a new structure to hold elements of the below matrix and describe the storage benefit of your newly designed structure compared to a traditional double array storage.
   b) Give the number of elements as a function of matrix rox/column size N.
   c) Give the location of a given matrix element u[i][j] as a function of matrix rox/column size N.

   (For simplicity, assume all stripe matrices are N by N matrices.)

2	0	3	0
4	0	3	0
5	0	-3	0
20	0	12	0

10 . Assume a new type of special matrix, named stripe matrix, which has non-zero values in every a^{th} column. Below is a NxN stripe matrix M(N, a) where a is 3.

For this matrix, we would like to store the values, and their coordinates of the values to save space on the main memory.

a) Give the space usage of such representation as a function of N and a.

b) Explain under what circumstances such representation will use less space than traditional double array storage.

0	0	2	0	0	3	0
0	0	4	0	0	3	0
0	0	5	0	0	-3	0
0	0	20	0	0	12	0

Solution : a > 2 will save space , since we save 3 values to store 1 value, so at most we should have n/3 items in the array to save space.

Hard

---

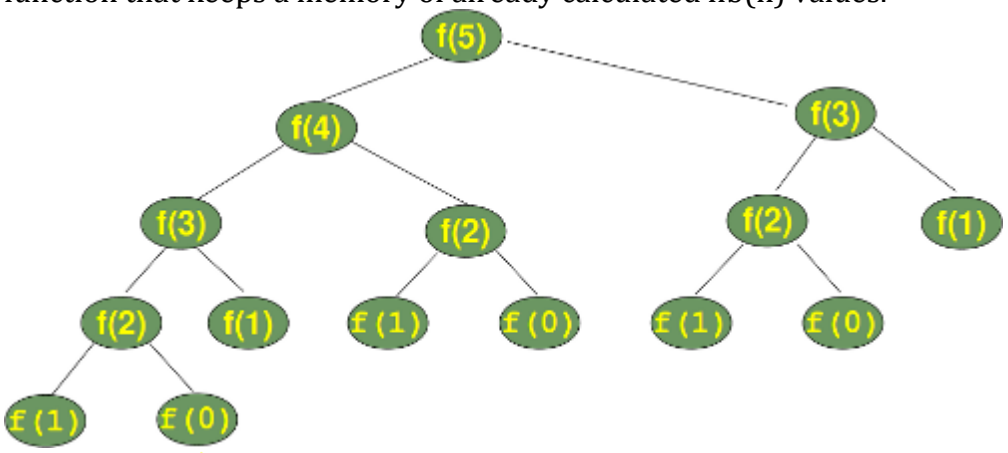
1. Explain the output and end result of below recursive call.

```
call(int n)
{
        print(n)
        call (n+1)
}
```

Solution : Recursive tracing goes infinitely. Stack overflow.

2. The problem with the recursive Fibonacci number solution was it keeps calculating the previously found values of lower Fibonacci sequences. For example, below diagram shows calculating Fib(5) where unnecessarily several same fibonacci values were calculated repeatedly .

For a solution, assume having an array of memory items, where we keep already calculated fib (n) values, then we won't be calculating the same fib(n) values during the intermediate calculation. Thus using this idea, implement a new recursive function that keeps a memory of already calculated fib(n) values.



Solution:

```
memo = {0:0, 1:1}
def fibm(n):
        if not n in memo:
                memo[n] = fibm(n-1) + fibm(n-2)
return memo[n]
```

3. Write an iterative and/or recursive function, which implements the Pascal's triangle:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

Solution:

```
def pascal(n):
        if n == 1:
                return [1]
        else:
```

```
            line = [1]
        previous_line = pascal(n-1)
        for i in range(len(previous_line)-1):
            line.append(previous_line[i] + previous_line[i+1])
        line += [1]
        return line
```

4. Write the output of below code segment? Assume the address of z[0][0][0] is 100.

Give a matrix

```
int z[4][2][6] ;
printf(" *(z[1]+1) = %d\n", * (z[1]+1));
printf(" *(z[1][1]+1) = %d\n", * (z[1][1]+1));
```

5.In the class we have seen the upper / lower triangular matrices. We calculated the number of items in lower triangular matrix which was n*(n+1)/2. For a given position u[i][j] we also showed that the address of u[i][j] is at i(i+1)/2 + j.
Do the same calculations for upper triangular matrix and give the number of items and the position of u[i][j].

Solution : number of items do not change

The position is 0 row has n items, 1st row has n-1 items, … i th row has n-i items. We need to add all items including ith row.

n + n -1 + n-2 + … + n-i = n*(n+1)/2 – i*(i+1)/2
= (n^2 + n – i^2 –i)/2

this is number of items before and including the ith row, we need to substract the items after jth position. There are n-1-j more elements on that row. So we need to substract n-1 –j and need to increment 1 for starting 0 index

= (n^2 + n – i^2 –i)/2 – (n-1-j) +1

Sample:

0	1	2	3
	4	5	6
		7	8
			9

i =2 j=2 n=4
16+4-4-2 / 2 – (4-1-2)
14/2 = 7 -1 =6

1. Given an unsorted linked list, and without using a temporary buffer, fill in the blanks in the below method that will delete any duplicates from the linked list. (Brute force solution is acceptable)

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

Answer:

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

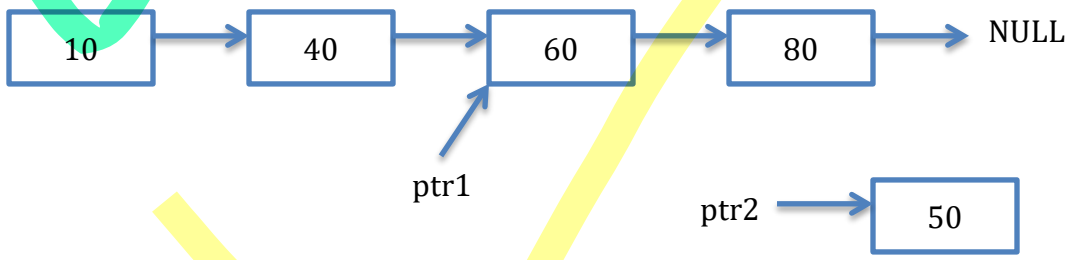
2. Given two sorted linked lists, merge the lists to form a single sorted linked list. Reuse one of the lists, Complete the given code below.

```
Node * Merge (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(_____) {
                if (_____ > _____){
                        temp->next = list2;
                        list2 = list2->next;
                }
                else {
                        temp->next = list1;
                        list1 = list1->next;
                }
                temp = temp->next;
        }
        if (list1 == null)
                _____;
        else
                _____;

        return newhead;
}
```

Answer:

```
Node * Merge (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;
}
```

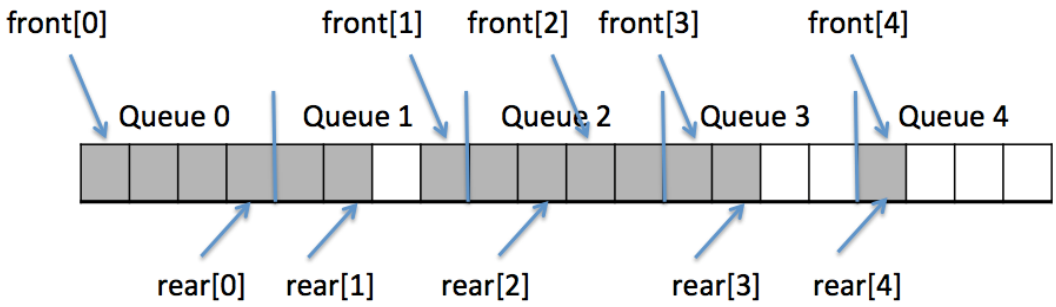
3. Consider the sorted linked list with these nodes:



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.
Correctly insert the new node to the above list in the sorted order. Explain your answer in sentences. No coding is required.

4. 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. Can you explain why postfix is more preferable than prefix?

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 multi queue is given below where there are 5 queues and in each of them there are 4 positions. Queue 0 and Queue 2 are full and others are not. For each queue a front and a rear pointer is kept.



Explain the steps that need to be taken when adding to a full queue and take recovery steps. Use short sentences. No coding is required.

Burada bu kadar birakabiliriz veya

Explain the steps that need to be taken when adding to a full queue and take recovery steps. Write only steps needed to update for the full query and its items inside. Use short sentences. No coding is required. (Hint there are 2 scenarios)

Veya

Ayni sekilde ve altta asagidaki gibi bosluklar

Step 1. Check the queues in the right side to find an empty space
Step 1.1 If found move the queues
Step 1.2 move data inside the full queue
Step 1.2.1. _____ (fill here)

Step 2 If no empty queue on the right, check the queues in the left side to find an empty space
Step 2.1 If found move the queues
Step 2.2 move data inside the full queue
Step 2.2.1. _____ (fill here)