CELEBRATING 25 YEARS
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 1 – Sep-2019								
Sub:	Data Structures and Applications					Sub Code:	18CS32	Branch: CSE
Date:	09/09/19	Duration:	90 mins	Max Marks:	50	Sem/ Sec:	3rd /A,B,C	OBE

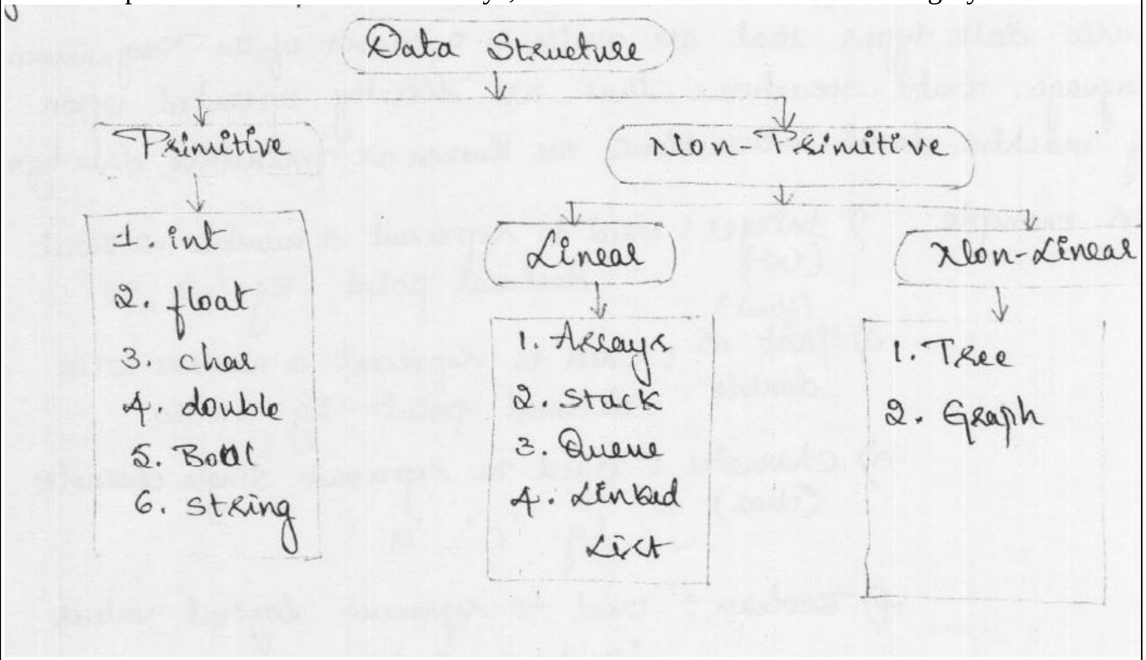
Answer any FIVE FULL Questions	MARK S	CO	RB T
1. Define Data Structures. Explain the different types of data structures with examples?	[1+2+2]	CO1	L1

Data structure is a representation of logical relationship existing between individual elements of data.In other words, a data structure defines a way of organizing all data items that considers not only theelements stored but also their relationship to each other. The term data structure is used to describethe way data is stored.

Data structures are divided into two types:
• Primitive data structures.
• Non-primitive data structures.

Primitive Data Structures are the basic data structures that directly operate upon the machine
instructions. They have different representations on different computers. Integers, floating pointnumbers, character constants, string constants and pointers come under this category.

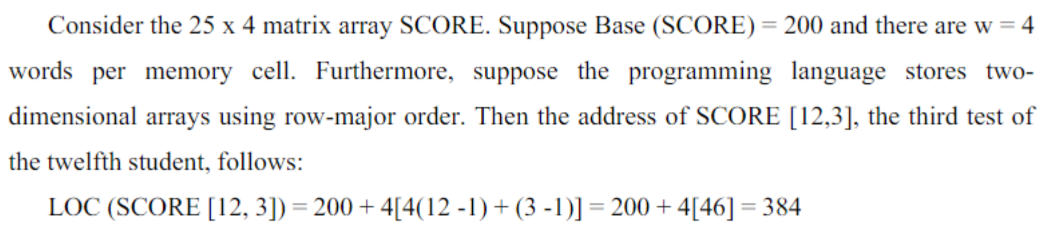
Non-primitive data structures are more complicated data structures and are derived from primitivedata structures. They emphasize on grouping same or different data items with relationship betweeneach data item. Arrays, lists and files come under this category.



1.bWhat is the formula to calculate the location in the row major and column major? Suppose each student in a class of 25 is given 4 tests, assume the students are numbered from 1 to 25, and the test scores are assigned in the 25 X 4 matrix called SCORE. Suppose Base (SCORE)=200, w=4, and the programming language uses column major order to store this 2D array, then find the address of 3rd test of 12th student i.e SCORE(12, 3).?	[2+2+1]	CO1	L4

Answer:

If we consider element in row r, column c in an array of M x N order, if B is the base address and w is the size of each element in the array, then the address of the elements can be given by

Row major address = B + w * (N * (r-1) + (c-1)) Column major address = B + w * (M * (c-1) + (r-1))  Consider the 25 x 4 matrix array SCORE. Suppose Base (SCORE) = 200 and there are w = 4 words per memory cell. Furthermore, suppose the programming language stores two-dimensional arrays using row-major order. Then the address of SCORE [12,3], the third test of the twelfth student, follows: LOC (SCORE [12, 3]) = 200 + 4[4(12 -1) + (3 -1)] = 200 + 4[46] = 384 LOC (Score[12,3])=200 + 4 (25 (3-1) + (12 -1)) = 444			
2**Write the Knuth Morris Pratt pattern matching algorithm and apply the same to search the pattern‘abcdabcy’ in the text ‘abcxabcdabxabcdabcdabcy’. Demonstrate steps also.**	[4+3+3]	CO1	L3

3a. **What is a pointer? Explain pointer declaration and initialization with syntax and example**
 	
Answer:
 	
Pointers in C language is a variable that stores/points the address of another variable. A Pointer in C is used to allocate memory dynamically i.e. at run time. The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.
 	
• Pointer Syntax : data_type *var_name; Example : int *p; char *p;
• Where, * is used to denote that “p” is pointer variable and not a normal variable.
 	
#include <stdio.h>
int main()
{
  int *ptr, q;
  q = 50;
  /* address of q is assigned to ptr */
  ptr = &q;
  /* display q's value using ptr variable */
  printf("%d", *ptr);
  return 0;
}
	[1.5+1.5+2]	CO1	L1

3b. Write a C program with an appropriate structure definition and variable declaration to store information about an employee, using nested structures. Consider the following fields like: ENAME, EMPID. DOJ (Date, Month, Year).	[2.5+2.5]	CO1	L3

Answer:

```
#include<stdio.h>

struct employee
{
        char ename[30];
        char empid[10];
        struct doj
        {
                int dd,mm,yy;
        }d;

}e;

int main()
{
        printf("\nEnter the details of the employee");
        scanf("%s%s%d%d%d", e.ename, e.empid, &e.d.dd, &e.d.mm, &e.d.yy);

        printf("\nThe details of the employee are: \n");
        printf("%s\n%s\n%d\n%d\n%d\n", e.ename, e.empid, e.d.dd, e.d.mm, e.d.yy);


        return 0;
}
```

4. **What is dynamic memory allocation? Explain different functions associated with dynamic memory allocation and deallocation with syntax and example. Code a C program to illustrate the same for allocating memory to store n integers and find the sum using dynamic memory allocation.**	5+2.5+2.5	CO4	L3

Answer:

Dynamic Memory Allocation can be defined as a procedure in which the size of a data structure (like Array) is changed during the runtime.
C provides some functions to achieve these tasks. There are 4 library functions provided by C defined under **<stdlib.h>** header file to facilitate dynamic memory allocation in C programming. They are:

1. malloc()
2. calloc()
3. free()
4. realloc()

Dynamic memory allocation

Dynamic memory allocation is an aspect of allocating and freeing memory according to your needs. Dynamic memory is managed and served with pointers that point to the newly allocated space of memory in an area which we call the **heap**. Now you can create and destroy an array of elements at runtime without any problems.

To sum up, the automatic memory management uses the stack, and the dynamic memory allocation uses the heap.

The <stdlib.h> library has functions responsible for dynamic memory management.

Function	Purpose
malloc	Allocates the memory of requested size and returns the pointer to the first byte of allocated space.
calloc	Allocates the space for elements of an array. Initializes the elements to zero and returns a pointer to the memory.
realloc	It is used to modify the size of previously allocated memory space.
Free	Frees or empties the previously allocated memory space.

```
// Program to calculate the sum of n numbers entered by the user

#include<stdio.h>
#include<stdlib.h>

int main()
{
int n, i, *ptr, sum = 0;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    ptr = (int*) malloc(n * sizeof(int));

// if memory cannot be allocated
if(ptr == NULL)
    {
        printf("Error! memory not allocated.");
exit(0);
    }

    printf("Enter elements: ");
for(i = 0; i < n; ++i)
    {
        scanf("%d", ptr + i);
        sum += *(ptr + i);
    }

    printf("Sum = %d", sum);

// deallocating the memory
    free(ptr);

return0;
}
```

5. **Define stack. Write a C program demonstrating the various stack operations, including cases foroverflow and underflow of stacks**	**2 * 5**	CO3	L3

A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: **push** the item into the stack, and **pop** the item out of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. **push** adds an item to the top of the stack, **pop** removes the item from the top. A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top.

```
#include<stdio.h>

int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
    //clrscr();
    top=-1;
    printf("\n Enter the size of STACK[MAX=100]:");
    scanf("%d",&n);
    printf("\n\t STACK OPERATIONS USING ARRAY");
    printf("\n\t--------------------------------");
    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
    do
    {
        printf("\n Enter the Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
            }
            case 2:
            {
                pop();
                break;
            }
            case 3:
            {
                display();
                break;
            }
            case 4:
            {
                printf("\n\t EXIT POINT ");
                break;
            }
            default:
            {
                printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
            }

        }
    }
    while(choice!=4);
    return 0;
}
void push()
{
    if(top>=n-1)
    {
        printf("\n\tSTACK is over flow");

    }
    else
    {
        printf(" Enter a value to be pushed:");
        scanf("%d",&x);
        top++;
        stack[top]=x;
    }
}
void pop()
{
    if(top<=-1)
    {
        printf("\n\t Stack is under flow");
    }
    else
    {
        printf("\n\t The popped elements is %d",stack[top]);
        top--;
    }
}
void display()
{
    if(top>=0)
    {
        printf("\n The elements in STACK \n");
        for(i=top; i>=0; i--)
            printf("\n%d",stack[i]);
        printf("\n Press Next Choice");
    }
    else
    {
        printf("\n The STACK is empty");
    }

}
```

6a. **Write a C function to evaluate postfix expression** [2.5+2.5] CO4 L3

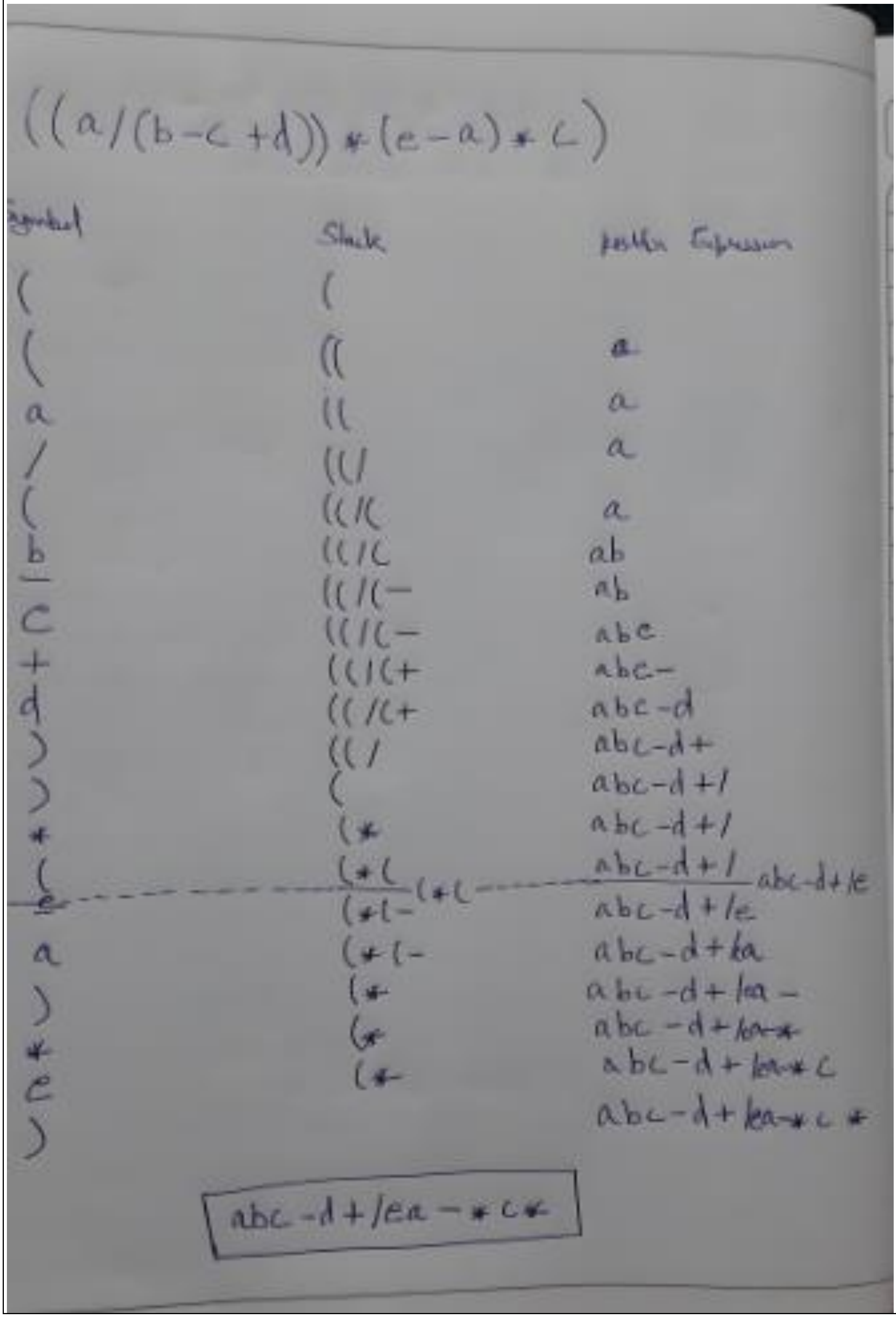
```
// The main function that returns value of a given postfix expression
intevaluatePostfix(char* exp)
{
    // Create a stack of capacity equal to expression size
    structStack* stack = createStack(strlen(exp));
    inti;

    // See if stack was created successfully
    if(!stack) return-1;

    // Scan all characters one by one
    for(i = 0; exp[i]; ++i)
    {
        // If the scanned character is an operand (number here),
        // push it to the stack.
        if(isdigit(exp[i]))
            push(stack, exp[i] - '0');

        // If the scanned character is an operator, pop two
        // elements from stack apply the operator
        else
        {
            intval1 = pop(stack);
            intval2 = pop(stack);
            switch(exp[i])
            {
            case'+': push(stack, val2 + val1); break;
            case'-': push(stack, val2 - val1); break;
            case'*': push(stack, val2 * val1); break;
            case'/': push(stack, val2/val1); break;
            }
        }
    }
    returnpop(stack);
}

// C program to evaluate value of a postfix expression
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

// Stack type
structStack
{
    inttop;
    unsigned capacity;
    int* array;
};

// Stack Operations
structStack* createStack( unsigned capacity )
{
    structStack* stack = (structStack*) malloc(sizeof(structStack));

    if(!stack) returnNULL;

    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (int*) malloc(stack->capacity * sizeof(int));

    if(!stack->array) returnNULL;

    returnstack;
}
```

```
intisEmpty(structStack* stack)
{
    returnstack->top == -1 ;
}

charpeek(structStack* stack)
{
    returnstack->array[stack->top];
}

charpop(structStack* stack)
{
    if(!isEmpty(stack))
        returnstack->array[stack->top--] ;
    return'$';
}

voidpush(structStack* stack, charop)
{
    stack->array[++stack->top] = op;
}



// Driver program to test above functions
intmain()
{
    charexp[] = "231*+9-";
    printf("postfix evaluation: %d", evaluatePostfix(exp));
    return0;
}
```

Or can also be

```
#include<stdio.h>
intstack[20];
inttop = -1;

voidpush(intx)
{
        stack[++top] = x;
}

intpop()
{
        returnstack[top--];
}

intmain()
{
        charexp[20];
        char*e;
        intn1,n2,n3,num;
        printf("Enter the expression :: ");
        scanf("%s",exp);
        e = exp;
        while(*e != '\0')
        {
                if(isdigit(*e))
                {
                        num = *e - 48;
                        push(num);
                }
                else
                {
                        n1 = pop();
                        n2 = pop();
                        switch(*e)
                        {
                                case'+':
                                {
                                        n3 = n1 + n2;
                    break;
                                }
                                case'-':
                                {
                                        n3 = n2 - n1;
                                        break;
                                }
                                case'*':
                                {
                                        n3 = n1 * n2;
                                        break;
                                }
                                case'/':
                                {
                                        n3 = n2 / n1;
                                        break;
                                }
                        }
                        push(n3);
                }
                e++;
        }
        printf("\nThe result of expression %s  =  %d\n\n",exp,pop());
        return0;

}
```

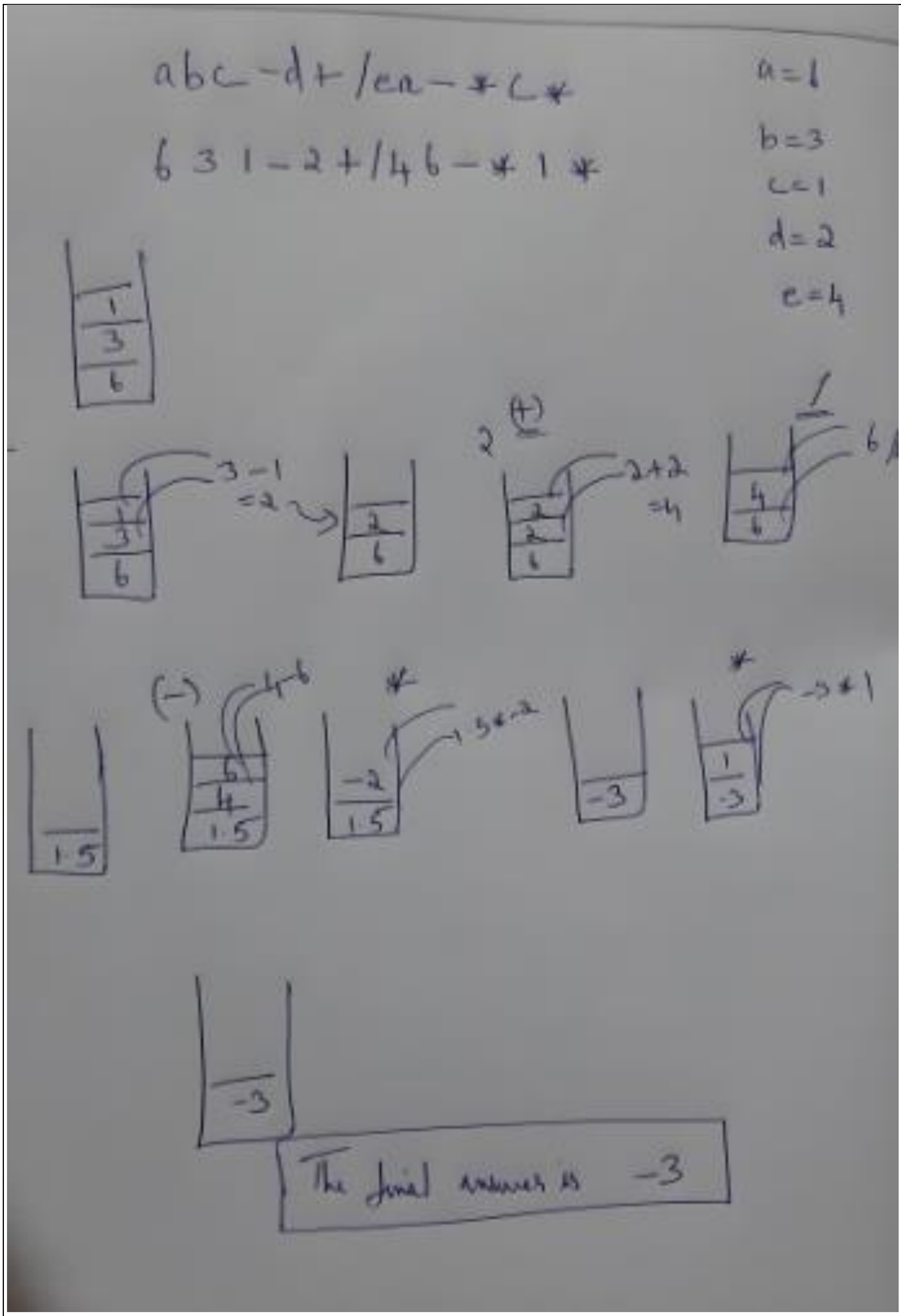
6b. **What is the output of the following code?**
int num[5] = { 3, 4, 6, 2, 1 };
int *p = num;
int *q = num + 2;
int *r = &num[1];
printf("%d %d", num[2], *(num+2));
printf("%d %d", *p, *(p+1));
printf("%d %d", *q, *(q+1));)
printf("%d %d", *r, *(r+1));)
 	
Answer:
 	
6 6
3 4
6 2
4 6
	[4+2]	CO4	L4

7. Convert the infix expression ((a/(b-c+d))*(e-a)*c) to postfix expression and evaluate that postfix expression for given data a=6, b=3, c=1, d=2, e=4 (using stack representation). 	[4]	CO3	L3

((a/(b-c+d))*(e-a)*c)

Symbol	Stack	Postfix Expression
(	(	
(	((	a
a	((	a
/	((/	a
(	((/(	a
b	((/(	ab
-	((/(-	ab
c	((/(-	abc
+	((/(+	abc-
d	((/(+	abc-d
)	((/	abc-d+
)	(	abc-d+/
*	(*	abc-d+/
(	(*(	abc-d+/
~~e~~	(*(	abc-d+/e
	(*(-	abc-d+/e
a	(*(-	abc-d+/ea
)	(*	abc-d+/ea-
*	(*	abc-d+/ea-*
c	(*	abc-d+/ea-*c
)		abc-d+/ea-*c*

$\boxed{abc-d+/ea-*c*}$


abc-d+/ea-*c*

6 3 1-2+/4 6-*1*

a=1
b=3
c=1
d=2
e=4

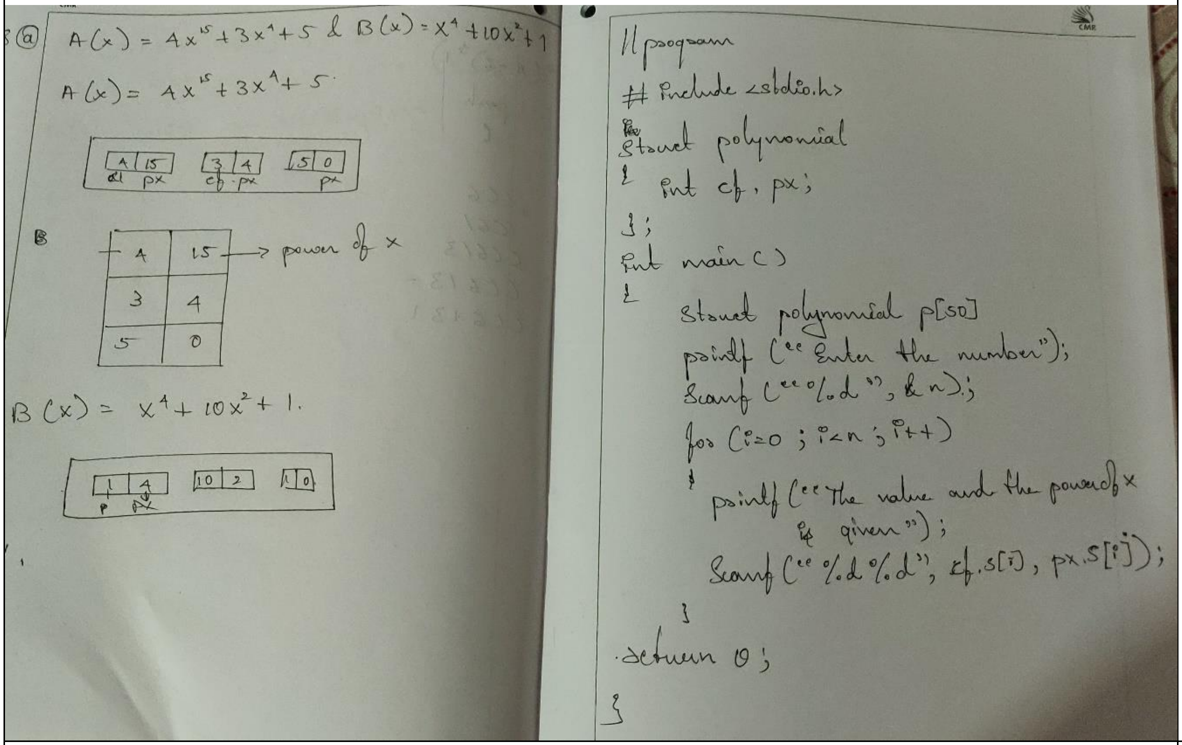
The final answer is -3

8a. Consider two polynomials,

$A(x) = 4x^{15} + 3x^4 + 5$ and $B(x) = x^4 + 10x^2 + 1$

Show diagrammatically how these two polynomials can be stored in a 1- D array. Also give its Crepresentation for initialization.

CO4 L3



8 (a) $A(x) = 4x^{15} + 3x^4 + 5$ & $B(x) = x^4 + 10x^2 + 1$

$A(x) = 4x^{15} + 3x^4 + 5$

4	15
3	4
5	0

cf px · cf · px · px

B → power of x

$B(x) = x^4 + 10x^2 + 1.$

1 4 · 10 2 · 1 0

p px

```
//program
#include <stdio.h>
struct polynomial
{  int cf, px;
};
int main ()
{
     struct polynomial p[50]
     printf ("Enter the number");
     scanf ("%d", &n);;
     for (i=0 ; i<n ; i++)
     {
       printf ("The value and the power of x
               is given");
       scanf ("%d %d", cf.s[i], px.s[i]);
     }
return 0;
}
```

8b. Write a C function to perform selection sort

CO1 L3

```
#include <stdio.h>

int main()
{
  int array[100], n, c, d, position, swap;

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  for (c = 0; c < (n - 1); c++)
  {
    position = c;

    for (d = c + 1; d < n; d++)
    {
      if (array[position] > array[d])
        position = d;
    }
    if (position != c)
    {
      swap = array[c];
      array[c] = array[position];
      array[position] = swap;
    }
  }
   printf("Sorted list in ascending order:\n");

  for (c = 0; c < n; c++)
    printf("%d\n", array[c]);

  return 0;
}
```