Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Practice Sheet #09 with Solutions

Topic: Structures in C

Date: 20-03-2017

1. Let $\omega = \sqrt[3]{2}$ be the real cube root of 2. Consider the set

$$A = \{a + b\omega + c\omega^2 \mid a, b, c \text{ are integers}\}$$

of real numbers. It turns out that the set A is closed under addition, subtraction and multiplication, i.e., the sum, difference and product of $a_1 + b_1\omega + c_1\omega^2, a_2 + b_2\omega + c_2\omega^2 \in A$ can be expressed in the form $a + b\omega + c\omega^2$. Clearly, $(a_1 + b_1\omega + c_1\omega^2) \pm (a_2 + b_2\omega + c_2\omega^2) = (a_1 \pm a_2) + (b_1 \pm b_2)\omega + (c_1 \pm c_2)\omega^2$.

For computing the product, first multiply $a_1 + b_1\omega + c_1\omega^2$ and $a_2 + b_2\omega + c_2\omega^2$ and obtain a polynomial in ω of degree 4. Then use the facts $\omega^3 = 2$ and $\omega^4 = 2\omega$ in order to reduce this polynomial of degree 4 back to a polynomial of degree 2. That is, we have:

$$\begin{aligned} & (a_1 + b_1\omega + c_1\omega^2)(a_2 + b_2\omega + c_2\omega^2) \\ = \; & (a_1a_2) + (a_1b_2 + a_2b_1)\omega + (a_1c_2 + b_1b_2 + a_2c_1)\omega^2 + (b_1c_2 + b_2c_1)\omega^3 + (c_1c_2)\omega^4 \\ = \; & (a_1a_2) + (a_1b_2 + a_2b_1)\omega + (a_1c_2 + b_1b_2 + a_2c_1)\omega^2 + (b_1c_2 + b_2c_1) \times 2 + (c_1c_2) \times (2\omega) \\ = \; & (a_1a_2 + 2b_1c_2 + 2b_2c_1) + (a_1b_2 + a_2b_1 + 2c_1c_2)\omega + (a_1c_2 + b_1b_2 + a_2c_1)\omega^2. \end{aligned}$$

Represent an element of A by a structure of three integers:

```
typedef struct {
   int a,b,c; /* Represents a + bω + cω² ∈ A */
} cubicNumber;
```

Complete the following function that takes two cubic numbers **x1**, **x2** as arguments and returns their product. **(8)**

```
cubicNumber cubicProd ( cubicNumber x1, cubicNumber x2 )
{
   ____________________________________ /* local variable */

   _______ = ____________________________________________________________;

   _______ = ____________________________________________________________;

   _______ = ____________________________________________________________;

   ____________________________________
}
```

2- Which languages necessarily need heap allocation in the runtime environment?
A. Those that support recursion
B. Those that use dynamic scoping
C. Those that allow dynamic data structures
D. Those that use global variables

3- The following C function takes a simply-linked list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank.

```
typedef struct node
{
 int value;
 struct node *next;
}Node;

Node *move_to_front(Node *head)
{
 Node *p, *q;
 if ((head == NULL: || (head->next == NULL))
   return head;
 q = NULL; p = head;
 while (p-> next !=NULL)
 {
   q = p;
   p = p->next;
 }
 _____________________________________
 return head;
}
```

Choose the correct alternative to replace the blank line.

A. q = NULL; p->next = head; head = p;
B. q->next = NULL; head = p; p->next = head;
C. head = p; p->next = q; q->next = NULL;
D. q->next = NULL; p->next = head; head = p;

4- Consider the following C program segment where CellNode represents a node in a binary tree:

```
struct CellNode
{
 struct CellNOde *leftChild;
 int element;
```

```
  struct CellNode *rightChild;
};

int GetValue(struct CellNode *ptr)
{
  int value = 0;
  if (ptr != NULL)
  {
   if ((ptr->leftChild == NULL) &&
       (ptr->rightChild == NULL))
     value = 1;
   else
     value = value + GetValue(ptr->leftChild)
                   + GetValue(ptr->rightChild);
  }
  return(value);
}
```

The value returned by GetValue() when a pointer to the root of a binary tree is passed asits argument is:

A. the number of nodes in the tree
B. the number of internal nodes in the tree
C. **the number of leaf nodes in the tree**
D. the height of the tree

5- Given a sequence a0,a1,...,an−1 of distinct numbers, the problem of All Nearest Smaller Value(ANSV)is to identify for each number ai, the smallest j, j > i, j 6 n−1, such that aj < ai. It is undefined if no such j exists. For example, in the sequence 8,9,7,5,6, (indexed as 0,1,2,3,4), the ANSV's are 2,2,3,u,u, where u denotes "undefined" (the ANSV for the last element is always undefined). Note that you cannot change the order of the input sequence, since ANSV's depend on this ordering.

(a) Determine the ANSV's for the sequence 7,2,3,8,1,5,4,9,6. (2)

(b) Complete the following program that computes the ANSV's by considering the elements from left to right and by keeping track of those elements for which the ANSV's are yet to be determined. Clearly, these elements form a monotonically increasing subsequence ai1,ai2,...,aik, ai1 < ai2 < ··· < aik, i1 < i2 < ··· < ik. Initially, the sequence consists of a0. In the j-th step (i.e., when aj+1 is considered), if aj+1 < aj (= aik) then the ANSV of aj is j + 1, otherwise we add aj+1 to the subsequence. More generally, aj+1 will be the ANSV for ais,ais+1,...,aik, if s > 1 and ais−1 < aj+1 < ais, or if s = 1 and aj+1 < ai1. Notice that ais can be detected by working backwards from the end of the sequence,

namely aik, and deleting them one by one. The new element is added at the end, i.e., the subsequence becomes ai1,ai2,...,ais−1,aj+1. Evidently, the subsequence can be maintained as a stack. When we have considered the entire array, the remaining elements in the stack are the ones whose ANSV's are undefined (use−1 to denote it). Note that in the stack, it is desirable to store the index of the elements (i.e., j instead of aj). We can recover aj from j.

```
#include <stdio.h>
main ()
{
int j, length, *a, *b, *stack, bottom, top;
printf("how many integers?\n");
scanf("%d", &length);
a = (int *)malloc(length*sizeof(int)); /* for storing the input */
b = (int *)malloc(length*sizeof(int)); /* for storing the ANSV's */ /* Allocate memory to
stack */
stack = _____;
for (j=0; j<length; j++) scanf("%d",&a[j]); /* Read input */
top = 0;
stack[top] = 0; /* Initialize the stack */
for ( _____; j+1<length; j++) { /* Consider array elements one-by-one */
        while( (top _____) && (a[j+1] < _____) ) { /* j+1 is the ANSV of all
elements in the stack that are larger than a[j+1] */
        b[stack[top]] = j+1;
_____;
}
top = top + 1;
stack[top] = _____;
}
for (; top _____; top--)_____ = -1; /* Undefined ANSV */
for (j=0; j<length; j++) printf("%d\n", b[j]); /* Print the ANSV's */
}
```

6- Assume that my machine supports 32-bit addresses. The structure myStructis declared as follows. What value is returned by sizeof(struct myStruct)on my machine?

```
struct myStruct
{
        long A;
        char B[10];
        float C[10]; struct myStruct *D;
}
```

(A) 16 (B) 22 (C) 58 (D) 88

7- Write a function that accepts as argument an array A of integers together with its size n and a non-negative integer k. The function should return another array, allocated dynamically within the function, which is obtained by cyclically shifting the input array A by k positions to the right. For example, upon the input of A = {2,4,6,1,3,9,5} of size n = 7 and k = 3, your function should return {3,9,5,2,4,6,1}.

```
int *rotateArray ( int A[], unsigned int n, unsigned int k )
{

}
```

8- Define a data type vector which consists of a dynamically allocated array to store the elements of a vector and an integer to designate the size (dimension) of the vector. Assume that we deal with vectors of floating-point numbers.

9- What is printed by the following program?

```
#include <stdio.h>
struct abc
{
 int a;
 int *b;
 int c[5];
 } ;
void foo (struct abc x, struct abc y[])
{
  x.a = 25;
 *(x.b) = 50;
 x.c[0] = 30;
 y[0] = x;
}
int main ()
{
 struct abc x, y[5] ;
 int n = 20;
 x.a = 5;
 x.b = &n;
 x.c[0] = 10;
 y[0] = x;
 foo (x, y) ;
 printf ("   x: %d, %d \n", x.a, x.c[0]) ;
 printf ("y[0]: %d, %d \n", y[0].a, y[0].c[0]) ;
 printf ("n=%d\n", n) ;
}
```

10- Define a structure customer to specify data of customer in a bank. The data to be stored is: Account number (integer), Name (character string having at most 50 characters), and Balance in account (integer).

11- Assume data for all the 100 customers of the bank are stored in the array :
struct customer bank[100];
The function, transaction, is used to perform a customer request for withdrawal or deposit to her account. Every such request is represented by the following three quantities: Account number of the customer, request type (0 for deposit and 1 for withdrawal) and amount. The transaction function returns 0 if the transaction fails and 1 otherwise. The transaction fails only when the account balance is less than the withdrawal amount requested.
The array bank (defined above) is another input to the transaction function and is suitably updated after every request. In case of a failed transaction no change is made in the bank array.
The header for the function transaction is given below, complete the body of the function.

```
int  transaction ( int account_number, int request_type, int amount,    struct customer
bank [ 100 ] ) {









}
```

12- Suppose we have the following statement inside a function foo( ), and int *p is a global variable:

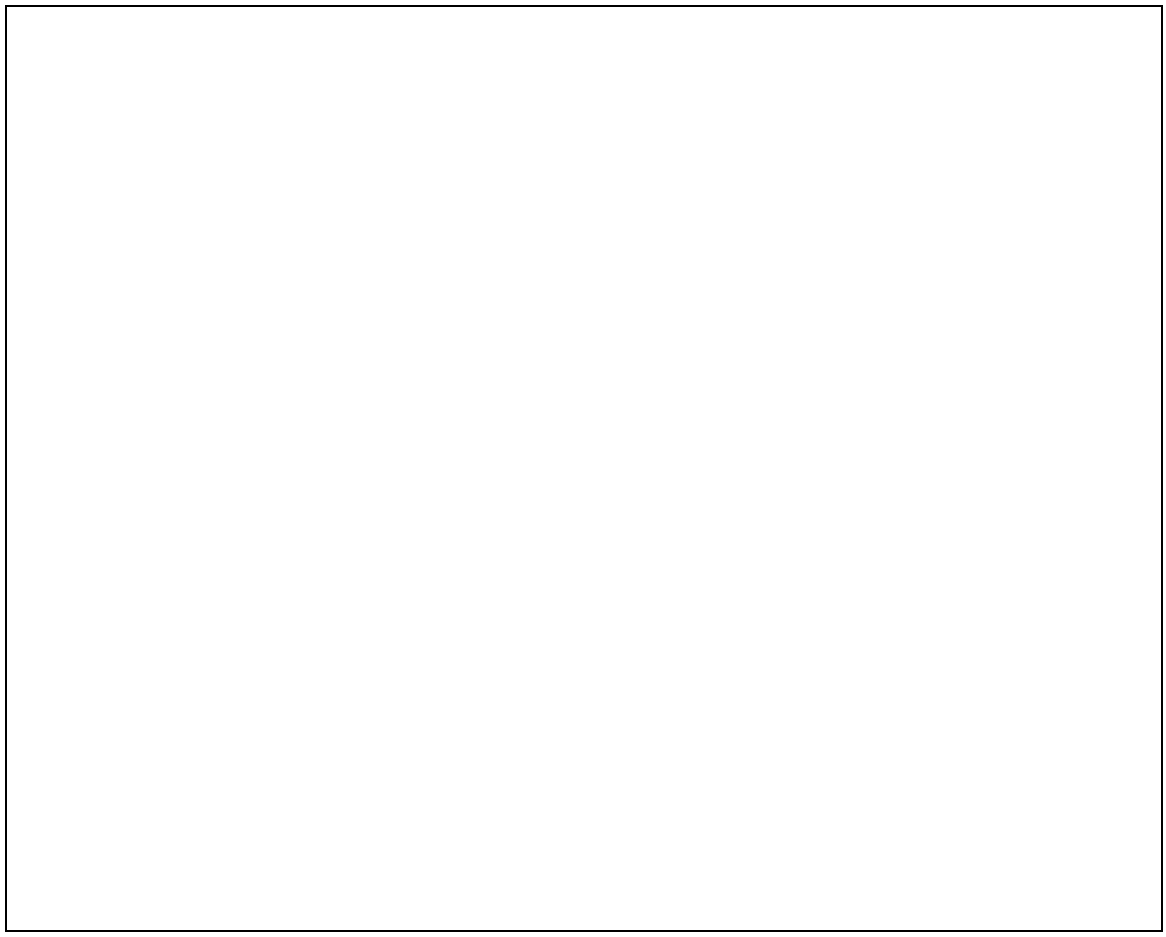
```
p = (int *) malloc(50 * sizeof(int));
```

Suppose main( ) calls foo( ). From which of the following places can we access the memory allocated by this call?
a. Only within the function foo( )
b. Within the function foo( ) and the function main( )
c. **Within any function, that is, the memory is global**

13- Define a structure ‘pt’ to represent a point in two dimensions. Use the pt structure to define a structure ‘rect’ representing a rectangle (in two dimensions).

Point	**Rectangle**

14- Write a function which takes as input an integer n, and an array of n rectangles, and returns the area of the smallest rectangle enclosing all the n rectangles in the array. The function prototype is as follows:
int bounding_rectangle (int n, struct rect rect_array[ ]);

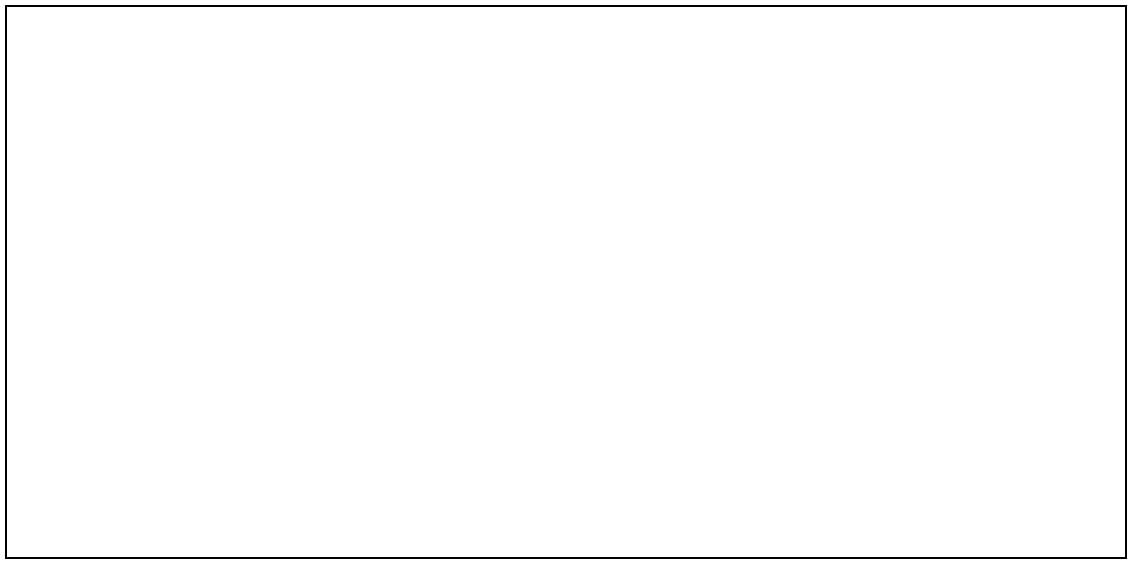


15- Consider the following code:

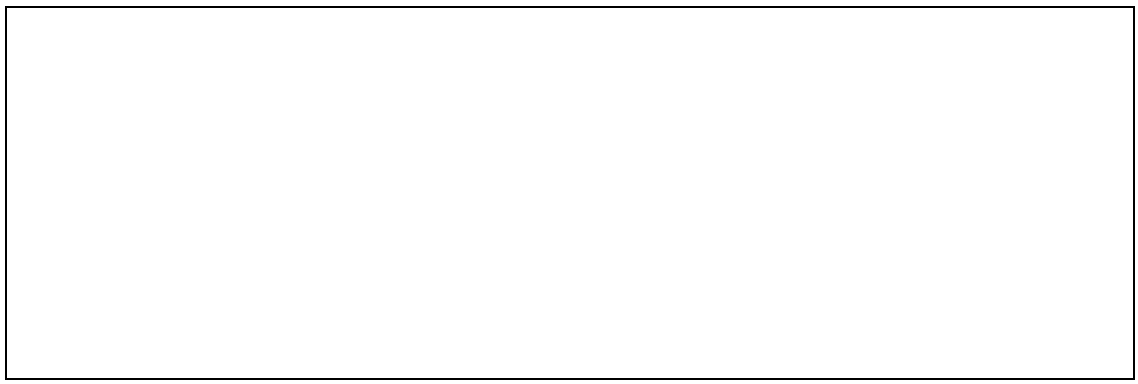
```
typedef  struct _Abc{
int emp_id;
char name[20];
char addr[25];
double salary;
} Abc ;

Abc *abc;
```

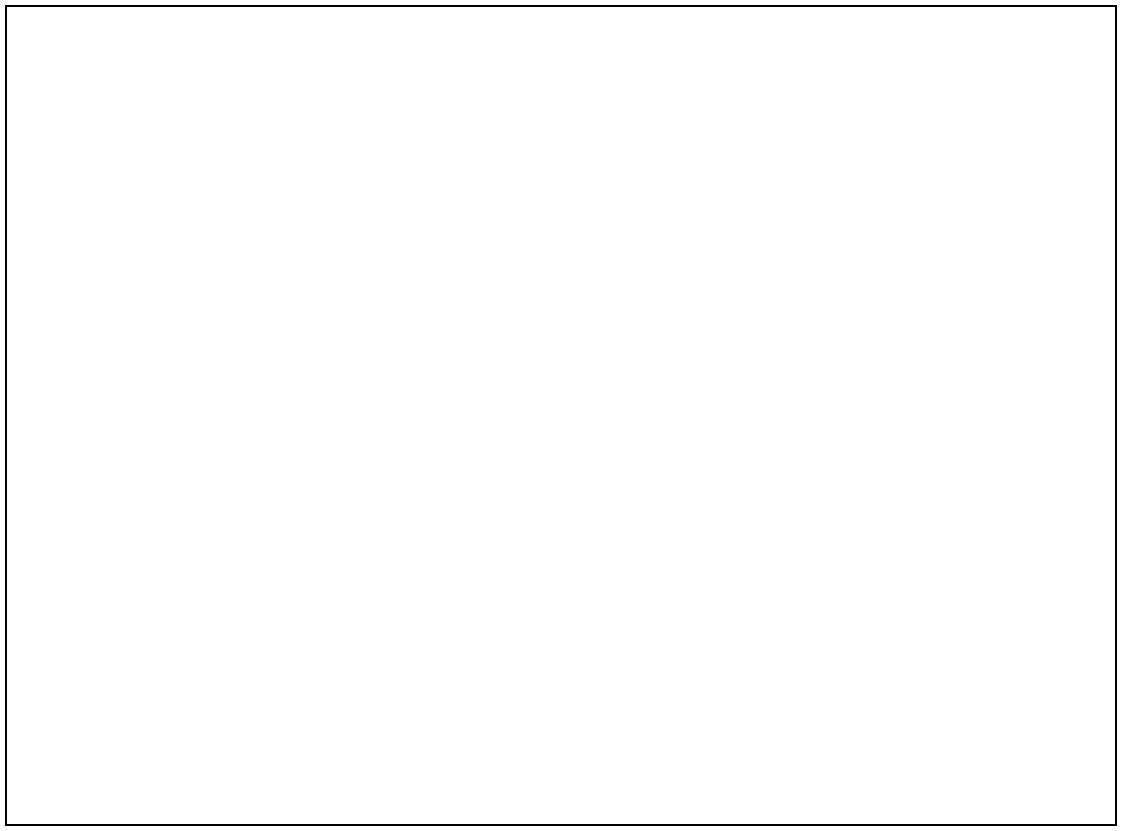
Assign a valid address to the pointer variable “abc” in the following two ways: (i) using an existing address of an appropriate variable, and (ii) by dynamically allocating the appropriate memory. Write code for printing all the data-fields which are pointed to by the pointer variable “abc”.



16- Declare a global array named **studArr** consisting of 600 **student** structures, where each structure **student** has the following members: **Name**: string of 10 characters, **Address**: string of 20 characters, **Roll_number**: integer, **CGPA:** float. Write the code to read the student data for 600 students from terminal.



17- There is a possibility that data corresponding to some students might have been entered multiple times. Write a function named elimDuplicate that takes no argument, and checks duplicate entries by matching the names in studArr. All those duplicate entries must be eliminated by moving forward the elements of studArr after a deleted entry. No additional array may be created for this purpose. After eliminating the duplicate student data, print out all the student data. [Hint: You may use any string library function, if you wish].



18- For the following entities, present the suitable structure declarations (struct in running lines):

a- The sinusoidal waveform $A\sin(\omega t + \delta)$:

---

b- Data for a student represented by his rollNnumber and name (both strings):

---

c- A course, represented by a courseNnumber (int) and data of a certain number (known at runtime) of students registered for the course:

19- A structure array struct ticket sold_tkt[1000] stores all the T (< 1000) tickets sold in a day. Assume, only one person travels in a single ticket and boards the train from the from station of the ticket. Also, the designated train of a ticket stops at the from and to stations of the ticket. Complete the function below which takes as input a train trn, and the sold_tkt[] array, and returns the number of passengers, with tickets, who boarded the train at an intermediate station (i.e., not at the starting station).

```
int boarded_on_way(int T, struct ticket sold_tkt[], struct train trn)
{









}
```

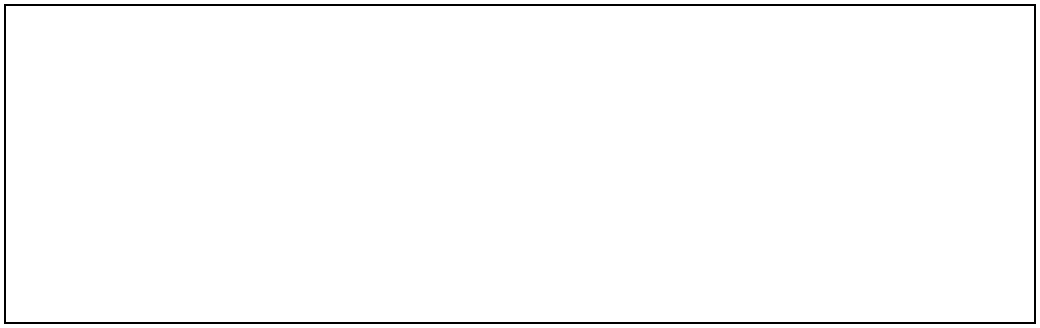
20- Many calendar systems e.g., Julian, Gregorian, Saka, Hijri etc are prevalent in different parts of the world. A calendar system can be represented by the number of months in a year, and an array of the number of days in each of these months. For example, in Gregorian calendar there are 12 months with 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, days respectively. A structure to represent a calendar system can be defined as:
struct calendar { int months; int days[15];};
A structure to represent a date (without year) can be defined as:
struct date {int dd; int mm;};

(i) Complete the function IsValidDate() which takes as input a date dt and a calendar system cs, and returns 1 if the date is valid according to the calendar system cs, or zero if otherwise.

(ii) Complete the function Convert( ) which takes as input a valid date dt1 in calendar system cs1 and returns it as a date dt2 in calendar system cs2. Assume that the start of years coincide in both calendar systems.

21- A robot lands on the surface of Mars and moves around to explore it. The robot's location can be represented by its (x, y) co-ordinates. The robot can move in one of the four directions in the co-ordinate system: left (L), right (R), up (U), and down (D) in a single move. Every single move changes position in the corresponding direction by 1 unit. In order to communicate its location to earth the robot transmits its initial location (x, y), and a character string representing its move sequence. For example, if the initial location of the robot is (1, 1) and the sequence of moves is "LURR", the new location is (2, 2). A 2-dimensional location is represented by the structure: struct location {int x; int y;};

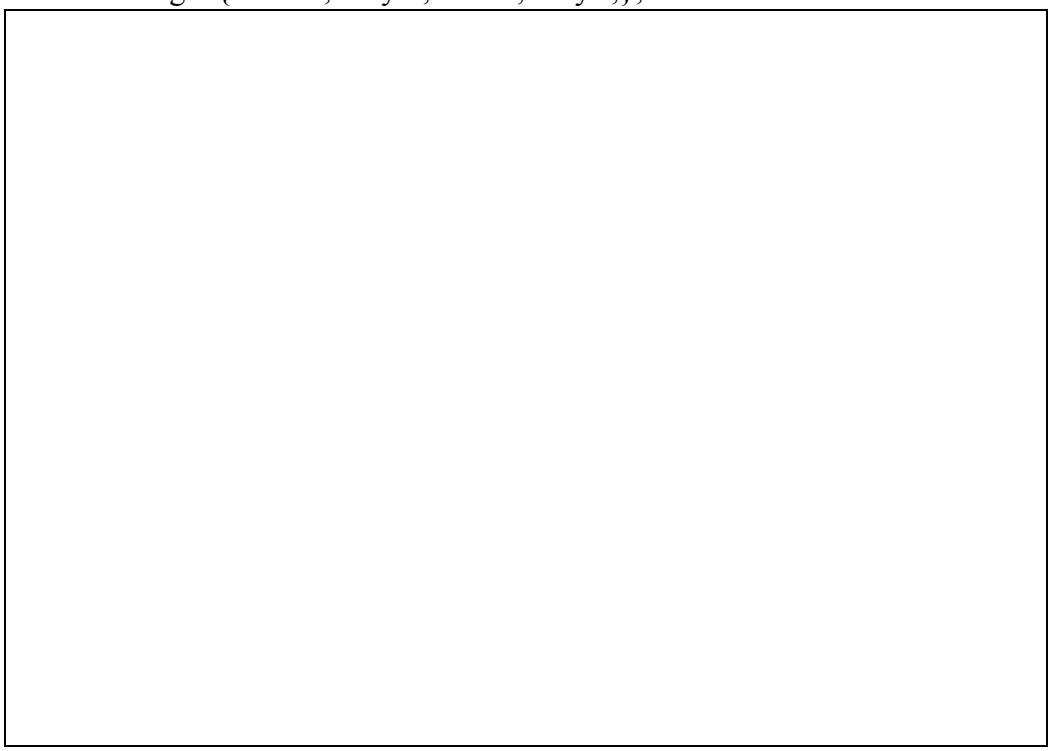
i- Complete the function newLocation(), which takes as input a location pointer locp, a single move (character L/R/U/D), and updates the location after executing the move.



ii- Suppose that the robot has lost communication and can be in any one of n locations. The space-ship radar now begins to search for it in a rectangular area. Complete the function boundingRectangle( ), which takes as input an array of n (< 10) locations and returns the smallest axis parallel rectangle which encloses all

these locations. A rectangle structure is defined in terms of the co-ordinates of its bottom left and top right corners as:

```
struct rectangle { int xbl; int ybl; int xtr; int ytr;};
```



iii- The robot was exploring Mars and communicating its move sequence to Earth. However, in the process of interplanetary communication a single character in the movement string was lost and had to be replaced by '?' (e.g., "LU?R"), where '?' might be any of the four movements. Finally, after a total of m (<100) moves, the robot exhausted its battery and became immobile. Since the final location of the robot is uncertain because of the '?' in the movement string, the space-ship radar has to search for it within an axis parallel search rectangle. Complete the function searchBox( ) below, which takes as input the initial location of the robot loc, a movement string moves (with a missing character replaced by '?'), total number of moves m (before getting immobile), and returns the smallest rectangle where the robot might be finally located.

22- Consider the stack data type: struct stack{ int data[MAXSIZE]; int top;}; Complete the following functions to (i) create an empty stack, (ii) check if a stack is empty, (iii) push into a nonfull stack, and (iv) pop from a non-empty stack. We define an empty stack to have top = -1.

