Exercise 1

Given the following sequence of pairs, where relation i-j indicates that vertex i is adjacent to vertex j:

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

Apply an on-line connectivity algorithm with weighted quickunion. Nodes are named with integers between 0 and 10. Show the contents of the array at each step and draw the final result as a forest.

Correct answer:

step 1: id 0 1 2 3 4 5 6 7 8 9 5 sz 1 1 1 1 1 2 1 1 1 1 1
step 2: id 3 1 2 3 4 5 6 7 8 9 5 sz 1 1 1 2 1 2 1 1 1 1 1
step 3: id 3 1 6 3 4 5 6 7 8 9 5 sz 1 1 1 2 1 2 2 1 1 1 1
step 4: id 3 1 6 3 5 5 6 7 8 9 5 sz 1 1 1 2 1 3 2 1 1 1 1
step 5: id 3 1 6 3 5 5 6 7 8 5 5 sz 1 1 1 2 1 4 2 1 1 1 1
step 6: id 3 1 6 3 5 5 6 7 5 5 5 sz 1 1 1 2 1 5 2 1 1 1 1
step 7: id 3 1 6 3 5 5 6 5 5 5 5 sz 1 1 1 2 1 6 2 1 1 1 1
step 8: id 3 5 6 3 5 5 6 5 5 5 5 sz 1 1 1 2 1 7 2 1 1 1 1

An answer is considered wrong if the replacement criterion of the weighted quick union approach is not satisfied (merging the smaller tree into the larger one).

Exercise 2

Sort the following integer array in ascending order by counting sort: $12_1\ 7_1\ 6_1\ 1\ 6_2\ 8_1\ 7_2\ 12_2\ 6_3\ 8_2\ 0\ 5\ 4\ 12_3$

Subscripts identify instances of the same key. Show the contents at each step of the intermediate data structures used.

Correct answer:

- k = 13. Data are in the range 0..k-1. The largest one is 12, thus k-1=12, k=13
- simple occurrences C: 1 1 0 0 1 1 3 2 2 0 0 0 3
- multiple occurrences C: 1 2 2 2 3 4 7 9 11 11 11 11 14
- final array: 0 1 4 5 6_1 6_2 6_3 7_1 7_2 8_1 8_2 12_1 12_2 12_3
- index on the final array of key 7_2: 8

Exercise 3

Given the following piece of code:

```
struct pack3
{
    int a;
};
struct pack2
{
```

```
    int b;
    struct pack3 *next;
};
struct pack1
{
    int c;
    struct pack2 *next;
};
struct pack1 data1, *data_ptr;
struct pack2 data2;
struct pack3 data3;
data1.c = 30;
data2.b = 20;
data3.a = 10;
data_ptr = &data1;
data1.next = &data2;
data2.next = &data3;
```

1) Per each of the following, say if the given expression is correct and, if so, which value is being accessed.
   1) `data1.c`
   2) `data_ptr->c`
   3) `data_ptr.c`
   4) `data1.next->b`
   5) `data_ptr->next->b`
   6) `data_ptr.next.b`
   7) `data_ptr->next.b`
   8) `(*(data_ptr->next)).b`
   9) `data1.next->next->a`

2) Which of the two variables `data2` and `data_ptr` occupies more space in memory? Why?

PROPOSED SOLUTION

1) Per each of the following expressions, say if it is correct and, if so, which value is being accessed.

EXPRESSION	CORRECT (Y/N)	VALUE
`data1.c`	Y	30
`data_ptr->c`	Y	30
`data_ptr.c`	N	
`data1.next->b`	Y	20
`data_ptr->next->b`	Y	20
`data_ptr.next.b`	N	

`data_ptr->next.b`	N	
`(*(data_ptr->next)).b`	Y	20
`data1.next->next->a`	Y	10

2) Which of the two variables `data2` and `data_ptr` occupies more space in memory? Why?

data2 occupies more memory. The total memory of data2 is the sum of the memory occupied by the int and t he pointer fields, while data_ptr is just a pointer.

Exercise 4

In the United States, telephone numbers are typically written in the form: (###)###-#### where # represents a 0 to 9 digit.

Write a C function `isValid` with the following prototype:

```
int isValid(char listOfNumbers[][M], int n);
```

where `listOfNumbers` is an array of strings and `n` is the number of telephone numbers that are stored in it, one per row. Assume that `M` is a constant equal to 14.
The function should check that each number in the given list is written in the correct format.
In case all the numbers are valid, the function should return +1. Otherwise, if even one number is not valid, it should return 0.

Proposed solution

```
int isValid(char listOfNumbers[][M], int n)
{
  int i,j;
  for (i=0;i<n;i++)
  {
    if (strlen(listOfNumbers[i])!=M-1)
      return 0;
    if (listOfNumbers[i][0]!='('||listOfNumbers[i][4]!=')'|| listOfNumbers[i][8]!='-')
      return 0;
    for (j=0;j<strlen(listOfNumbers[i]);j++)
      if(((j>0 && j<4)||(j>4 && j<8)||j>8) && !isdigit(listOfNumbers[i][j]))
          return 0;
  }
  return 1;
}
```

Exercise 5

Write a C function with the following prototype:

```
int findOddSubSequence(int vet[], int n);
```

where `vet` is an array of integers and `n` the number of stored elements. The function should compute the longest subsequence of `vet` with only odd values. The function should return the index position of the first occurrence of such subsequence in the array, or -1 if the subsequence does not exist.

For example, if `vet[]={9,3,2,3,3,2,3,5,7,1,7,4,3,9,15,1,7}`, the function should return the value 6, that is the index position of the first element of the subsequence highlighted in grey. Note that there is another subsequence with the same number of odd values, starting at index 12, but the function is required to return the position of the first occurring one.

Proposed solution

```
int findOddSubsequence(int vet[],int n)
{
  int i,l,lmax=-1,result=-1;
  for (i=0;i<n;i++)
  {
    if (vet[i]%2!=0)
      l+=1;
    else
    {
      if (l>lmax)
      {
        result = i-l;
        lmax = l;
      }
      l = 0;
    }
  }
  return result;
}
```

Exercise 6

A minefield is schematically represented by a matrix of characters, where the '*' character represents a mine and the '-' character represents a safe stepping spot.
Write a C function that looks for a path to safely cross the field from left to right, by stepping on adjacent safe spots. Make the following assumptions.

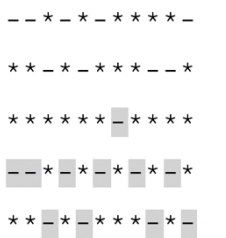
- The starting point of a safe path can only be in the first column of the matrix.
- You can only move left to right, from a safe spot to another one that is placed in the next column.
- Each move can happen either horizontally in the same row, or diagonally in the row before or after.
- At each step there can be no more than one safe spot (either on the same row, in the row before or in the row after). So, you can assume no branches in the path to be explored.

The function should have the following prototype:

```
int findPath(char minefield[][M],int nr, int nc);
```

where `minefield` is the given matrix of characters and `nr` and `nc` the number of rows and columns of the minefield, respectively. The function should return a `0` to `nr-1` value corresponding to the starting point (row index) of the safe path, if it exists, or `-1` if such path does not exist. In case there is more than one possible safe path, the function should return the lowest row index.

For example, if the minefield is the following:



the safe path is the one starting at row index 3 (highlighted in green). Hence, the function should return the value 3.

Proposed solution

```
int findPath(char field[][M],int nr,int nc)
{
  int i,j,rStep,r,start=-1;
  int found = 0; // flag that is set to 1 when a safe path is found

  for(i=0;i<nr && !found;i++)
  {
    if (field[i][0]== '-')  // possible starting point in the first column
    {
      found = 1;
      start = i;            // keep track of the current starting point
      rStep = i;
      for(j=1;j<nc && found; j++)  // explore the next columns
      {
        for(found = 0,r=rStep-1;r<=rStep+1 && !found; r++) // look for a safe stepping spot
        {
          if (r>=0 && r<nr && field[r][j]== '-') // a safe stepping spot is found
          {
            found = 1;
            rStep = r;                          // update the row index position
          }
        }
      }
    }
  }
  if (!found)
    return -1;
  else
    return start;
}
```