C++ Practice Exercises with solutions

Control Structures

1. Consider the following program:

```
int x,y;

cin >> x >> y;

while(x!=0 && y!=0){

        if(x>y){

                x = x-y;

        }else if(x<y){

                y = y-x;

        }else y=0;

}

cout << x+y << endl;
```

What will this program print on the following inputs:

a. 12 5

b. 27 39

c. -2 12

d. 23 0

2. Write a function that takes as input a positive integer n and returns the n-th harmonic number. Reminder: the n-th harmonic number is equal to 1+(1/2)+(1/3)+(1/4)+...+(1/n)

Solution:

```
double harmonic(int n)

{

        int i; double res=0.0;

        for(i=1; i<=n; i++)
        res += 1.0/i; //Note: not 1/i → this rounds down to 0
    return res;
}
```

3. Recall that the % operator calculates the integer remainder of a division (e.g. 8%3 is equal to 2). Consider the following code:

```
void hail(int n)
{
    if(n<=1) return;
    cout n << “ -> ” ;
    if(n%2) hail(3*n+1);
    else hail(n/2);
}
```

What will this function print for n=5? For n=17? For n=6?

Arrays

4. Write a function which will be given as input an array, its size and an integer p. The function will then cyclically shift the array p positions to the right: each element is moved p positions to the right, while the last p elements are moved to the beginning of the array. For example: if we have the array [1 2 3 4 5 6], shifting 2 positions to the right should give the array [5 6 1 2 3 4]. Your function should work correctly for negative values of p.

Solution:

```
void shift_right(int *A, int size, int p)
{
    int i;
    int tmp[size];
    p = p%size;
    for(i=0;i<size;i++){
        tmp[ i+p>=0? (i+p)%size : size+i+p ] = A[i];
    }
    for(i=0;i<size;i++) A[i] = tmp[i];
}
```

5. Write a function that decides if a given char array is a palindrome. A palindrome is a word/phrase that can be read the same from left to right as from right to left. Example: EVE, MADAMIMADAM, ABBA are palindromes.

Solution:

```
bool is_palindrome(char *word, int size)
{
    int i,j;
    for(i=0,j=size-1; i<j; i++,j--){
        if(word[i]!=word[j])
            return false;
    }
    return true;
}
```

6. Write a function which, given an array of integers, returns the integer that appears most frequently in the array. E.g. for the array [1 2 3 2 3 4 2 5] your function should return 2.

Solution:

```
int most_common(int *A, int size)
{
    int winner=A[0], freq=0;
    int i,j, tmp;

    for(i=0;i<size;i++){
        for(tmp=j=0; j<size ; j++) //count this item's frequency
            if(A[i]==A[j]) tmp++;
        if(tmp>freq){
            freq = tmp;
            winner = A[i];
        }
    }
    return winner;
}
```

Pointers and Memory Management

7. What will the following program print?

```
    int a[5] = { 1,2,3,4,5} ;
    int *p = a+2;
    int *q = new int[3];
    int i;
    for(i=0;i<3;i++)
        *q++ = *p++;
    cout << "Loop 1" << endl;
    while(p!=a){
        p--;
        cout << p[0] << endl;
    }
    cout << "Loop 2" << endl;
    for(i=1;i<=3;i++)
        cout << q[-i] << endl;
    q=a+1;
    cout << "Loop 3" << endl;
    for(i=0;i<4;i++)
        cout << q[i] << endl;
```

8. The program of exercise 7 contains a memory leak. Where is it? How can it be fixed?

Solution: Add the command **delete [ ] (q-3);** before the statement **q = a + 1;**

9. Write a function that is given two positive integers r,c and allocates a 2-dimensional array with r rows and c columns. The function then returns a pointer to the allocated array.

Solution:

```
int **get_2d_array(int r, int c)
{
        int **res = new int *[rows];
        int i;
        for(i=0; i<r; i++)
                res[i] = new int[c];
        return res;
}
```

Linked Lists

For the following exercises recall the definition of a basic linked list node:

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

10. Write a function which, given a pointer to the first element of a linked list returns true if all elements of the list are unique. In other words, the function should return false if and only if there exist two nodes in the list with the same data.

Solution:

```
bool all_unique(struct Node * head)
{
    while(head!=0){
        struct Node *tmp;
        for(tmp = head->next; tmp!=0; tmp=tmp->next)
            if(tmp->data==head->data)
                return false;
        head = head->next;
    }
```