Solution to Practice questions

Q.1 (25 points) Write a function `strrev` which takes in a string and returns the reverse of that string.

```c
char * strrev(char * s)
/* returns the reverse of the string s */
{
    int n;
    char * t;

    if (!s) return NULL;

    for (n=0; s[n] != 0; n++); // compute length of s

    t = new char[n+1];

    for (int i=0; i<n; i++)
        t[i] = s[n-1-i];

    t[n] = 0;
    return t;
}
```

Time complexity: $O(n)$, Space complexity: $O(n)$

Q.2 (25 points) Write a recursive binary search function to search a given sorted array of integers for a particular element. What is the time and space complexity of your algorithm?

```c
int binsearch(const int x, const int A[], const int left, const int right)
/* returns i such that A[i] = x if such an i exists, -1 otherwise;
   left and right gives the range in which the search is performed */
```
{ 
    if (right < left) return -1;
    int mid = (left+right)/2;
    if (A[mid] == x) return mid;
    if (x < A[mid]) return binsearch(x, A, left, mid-1);
    else return binsearch(x, A, mid+1, right);
}

Time complexity: O(log n), Space complexity: O(log n)

Q.3 (25 points) Write a function that takes in two sorted arrays A and B of sizes m and n respectively and returns a sorted array C of size m+n which contains elements from A and B. You need to free up the space allocated to A and B, and allocate the space for C. What is the time and space complexity of your algorithm?

int * combine(int A[], const int m, int B[], const int n) 
{
    int ap=0, bp=0, cp=0;
    int * C;

    C = new int[m+n];

    while((ap<m) && (bp<n))
        else C[cp++] = B[bp++];

    while (ap<m) C[cp++] = A[ap++];

    while (bp<n) C[cp++] = B[bp++];

    delete [] A; delete [] B;
    return(C);
}

Time complexity: O(m+n), Space complexity: O(m+n)