```
void processNums(int nums[], int n)
{
      int i=-1, j=0;
      for(;j<n; ++j){
           if(nums[j]==1){
            swap(nums[i+1],nums[j]);
                     ++i;
           }
      }

}
```

```
int main()
{
int nums[]={0, 1, 0, 0, 1, 1, 1, 0, 1, 0};
int n = 10;
cout << "Original array: ";
for (int i=0; i < n; i++)
          cout << nums[i] <<" ";
processNums(nums, n);
cout <<"\nProcessed array: ";
for (int i=0; i < n; i++)
          cout << nums[i] <<" ";

 return 0;

}
```

Output:

(b) Print the output of the following code:

```
void UpdateStr(char str[])
{
          for (int i = 1; str[i] != '\0'; i++){
                    if (str[i] >= 'a' && str[i] <= 'z' && str[i-1] ==' ')
                              str[i] = str[i] - 32;
          }
}
```

```
int main() {

          char str[] = "I am a student of programming!" ;
          UpdateStr(str);
          cout << str << endl;
}
```

Output:

```
int main()
{
    int P[20];
    P[0] = 0;
    P[1] = 0;

    for(int i = 2; i < 20; i++)
        P[i] = i;

    for(int i = 2; i< 5; i++){

        for( int j = 2*i; (P[i] != 0) && (j< 20); j+=i)
            P[j] = 0;
    }

    for(int i = 0; i< 20; i++){
        if( P[i] != 0)
            cout<< P[i]<<endl;
    }
    return 0;
}
```

Problem 2 [10 pts] Give the output of the following code:

```
int check(float& a, float b, float c)
{
    float r = (a / b * 100.0 + 0.50);
    int s = (int) r;
    a = s / 100.0;
    if (a - c >= 0)
    return 1;
    return 0;
}
```

```
int main(){
  float dat[5] = {750,740,755,745,730};
  for(int i=0; i < 5; i++)
  {
    cout << i << ":";
    if( check(dat[i],1000,0.75) )
      cout << dat[i];
    cout << endl;
  }
}
```

Problem 3 [15 pts] Write a C++ function called mergeArrays. It should accept three integer arrays: A, B and C, and two integers m and n representing the sizes of the first two arrays. It can be assumed that the size of C is m+n. Arrays A and B contains integers that are already sorted in the ascending order (i.e. increasing order). Your function must combine all the numbers in A and B and store them C in such a way that C is also sorted in the ascending order. Here is an example to clarify the requirements. In this case, the sizes of A and B are 6 and 4 respectively.

A	0	4	5	5	7	9				
B	-1	0	3	6						
C	-1	0	0	3	4	5	5	6	7	9

Note: You can use as many loops as you like, but this task should be accomplished *without* the use of **nested loops**. In particular, you should not apply any sorting algorithm to this problem. You cannot use or create any other function.

Solution

```
void mergeArrays(int A[], int sizeA, int B[], int sizeB, int
C[])
{
     int i = 0, j = 0, k = 0;
     while(i < sizeA && j < sizeB) {
           if(A[i] <= B[j]){
                 C[k++] = A[i++];
           }
           else{
                 C[k++] = B[j++];
           }
     }
     if(i == sizeA){
           while(j < sizeB){
                 C[k++] = B[j++];
           }
     }
     if(j == sizeB) {
           while(i < sizeA){
                 C[k++] = A[i++];
           }
     }

}
```

13) What will be the output of the following code? **[3]**

```
void function(int b[][3])
{
        ++b;
        b[1][1] = 9;
}


int main()
{
        int a[3][3] = { { 1,2,3} ,  { 4,5,6} , {7,8,9} };
        function(a);
        cout << a[2][1];
        return 0;
}
```


17) What will be the output of the following code? **[4]**

```
int main ( )
{
char *s[4] = {"black", "white", "yellow", "violet"};

        cout<<(*(s+1)+2)<<endl;
        cout<<*(*(s+2)+3);
return 0;
}
```

18) What is the value of Arr[7][9]; in the sample code below? **[4]**

```
void main()
{
        int i, j;
        int counter = 0;
        int Arr[8][9];
        for (i = 0; i < 9; i++)
                for (j = 0; j < 8; j++)
                {
                        Arr[j][i] = counter;
                        ++counter;
                }
        cout << Arr[7][8];

}
```

```
int list[5]={2,4,8,10,-1};
int nextList[5]={3,-1,0,1,-1};
int start = 2;
int Free = 4;
void magic(int val , int position){
    int start = ::start;
    for(int i = 0 ; i< position - 1 ; i++)
        start=nextList[start];
    list[Free]=val; nextList[Free]=nextList[start];
   nextList[start]=Free++;
}
void magic(){
    int start = ::start;
    while(start != -1){
        cout<<list[start]<<"->";
        start=nextList[start];
    }
    cout<<"*"<<endl;
 }
 int main()
{
    magic();
    magic(5,2);
    magic();
    return 0;
 }
```

Output:

8->2->10->4->*
8->2->5->10->4->*