1) Evaluate the following data declarations and expression.

```
double a, s = 4.5, q = 7.0, r = 2.0;
int k = 8, m = 11;

if (!(s*2 <= k+r) && q > r*3.0) a = (int) (k+s); else a = s + m % 8;
```

Only one of the following statements is true. Which one is it?

a) a equals 12.5
b) a equals 5.5
c) a equals 12.0
d) a equals 7.5

2) Consider the following function

```
int foo(char *s1, char *s2)
{
   int c=0, s, p, found;

   for (s=0; s1[s] != '\0'; s++)
   {
      for (p=0, found=0; s2[p] != '\0'; p++)
      {
         if (s2[p] == s1[s])
         {
            found = 1;
            break;
         }
      }
      if (!found) c++;
   }
   return c;
}
```

If we were to make the following call to `foo` what value would it return?

```
foo("Asia Pacific", "aeiou");
```

a) 5
b) 6
c) 7
d) 8

3) Which one of the following statements correctly describes the purpose of the function `foo` in the previous question?

   a) Count the all the characters in s1 that are not found in s2.
   b) Count the all the characters in s2 that are not found in s1.
   c) Count the all the characters that are common to both s1 and s2.
   d) Count the all the characters in s2 that are found in s1.

4) What is the efficiency of the function `foo`?

   a) $O(n!)$
   b) $O(n^2)$
   c) $O(n\lg_2 n)$
   d) $O(n)$

5) The `strcmp` function receives two strings `s1` and `s2`. If `s1` comes before `s2` alphabetically it returns –1, if it comes after `s2` it returns a 1 and if `s1` and `s2` are the same it returns 0.
   Study the following four pieces of code. Only one correctly implements `strcmp`. Which one is it?

```
a) int strcmp(char *s1, char *s2)
   {
      while (s1 == s2)
      {
         if (*s1 == ‘\0’) return 0;
         s1++;
         s2++;
      }
      if (s1 < s2) return -1;
      else return 1;
   }
```

```
b) int strcmp(char *s1, char *s2)
   {
      while (*s1 != *s2)
      {
         if (*s1 == ‘\0’) return 0;
         else if (*s1 < *s2) return -1;
         else return 1;
         s1++;
         s2++;
      }
   }
```

```
c) int strcmp(char *s1, char *s2)
   {
      for ( ; *s1 == *s2; s1++, s2++)
      {
         if (*s1 == ‘\0’) return 0;
      }
      if (*s1 < *s2) return -1;
      else return 1;
   }
```

```
d) int strcmp(char *s1, char *s2)
   {
      for ( ; *s1 == *s2; s1++, s2++)
      {
         if (*s1 == ‘\0’) return 0;
         else if (*s1 < *s2) return -1;
         else return 1;
      }
   }
```

6) A doubly linked list makes use of the following `struct` and `class`.

```
template <typename dataType> struct dnode
{
   dataType data;
   dnode *prev, *next;

   // constructors, destructors and other functions, including

   dnode(const dataType& dataItem, dnode *prevPtr, dnode *nextPtr) :
      data(dataItem), prev(prevPtr), next(nextPtr) {
   }
};

template <typename dataType> class dlist
{
   private:
      dnode<dataType> *head;   // points to first item in list
      dnode<dataType> *tail;   // points to last item in list
      int numItems;

   public:

   // constructors, destructors and other functions, including

   void pop_front()
};
```

Only one of the following four functions correctly implements the pop_front function. Which one is it?

a)

```
void pop_front()
{
   if (head == NULL) return;

   dnode<dataType> *removeNode = head;

   head = head->next;
   tail = tail->prev;
   head->prev = NULL;

   delete removeNode;
   numItems--;
}
```

b)

```
void pop_front()
{
   if (head == NULL) return;

   dnode<dataType> *removeNode = head;
   if (head != NULL) {
      head = head->next;
      head->prev = NULL;
   }
   else {
      tail = NULL;
   }
   delete removeNode;
   numItems--;
}
```

c)

```
void pop_front()
{
   if (head == NULL) return;

   dnode<dataType> *removeNode = head;

   head = head->next;
   head->prev = NULL;

   if (head == NULL) {
      tail = NULL;
   }

   delete removeNode;
   numItems--;
}
```

d)

```
void pop_front()
{
   if (head == NULL) return;

   dnode<dataType> *removeNode = head;
   head = head->next;
   if (head == NULL) {
      tail = NULL;
   }
   else {
      head->prev = NULL;
   }
   delete removeNode;
   numItems--;
}
```

7) What is the efficiency of removing an item from a linked list?

a) $O(1)$
b) $O(n^2)$
c) $O(n \lg_2 n)$
d) $O(n)$