Solutions Exam Imperative Programming (3 Nov. 2021)

Problem 1: Assignments (20 points)

1.1
```
/* x == y*z */
.....
/* x == y*z */

-->(a) y--; x = x - z;
   (b) z--; x = x + y;
   (c) y++; z--;

     /* x == y*z */
     /* x - z == (y-1)*z */
   y--;
     /* x - z == y*z */
   x = x - z;
     /* x = y*z*/
```

1.2
```
/* 3*x + 5*y == A */
.....
/* y == A */

   (a) y = (y - 3*x)/5;
-->(b) x = x + y; y = 3*x + 2*y;
   (c) y = (A - 3*x)/5;

     /* 3*x + 5*y == A */
     /* 3*(x + y) + 2*y == A */
   x = x + y;
     /* 3*x + 2*y == A */
   y = 3*x + 2*y;
     /* y == A */
```

1.3
```
/* x*x <= A < (x+1)*(x+1) */
.....
/* x <= A < x + y */

-->(a) y = 2*x + 1; x = x*x;
   (b) x = x*x; y = 2*x + 1;
   (c) y = 2*x - 1; x++;

     /* x*x <= A < (x+1)*(x+1) */
     /* x*x <= A < x*x + 2*x + 1 */
   y = 2*x + 1;
     /* x*x <= A < x*x + y */
   x = x*x;
     /* x <= A < x + y */
```

1.4
```
/* x == A, y == B */
x = x - y; y = x + y; x = 2*x - y;
.....

-->(a) /* x == A - 2*B, y == A */
   (b) /* 2*x == B, 2*y == A - B */
   (c) /* x == 3*A - 2*B, y == A + B */

     /* x == A, y == B */
     /* x - y == A - B, y == B */
   x = x - y;
     /* x == A - B, y == B */
     /* x == A - B, x + y == A */
   y = x + y;
     /* x == A - B, y == A */
     /* 2*x - y == A - 2*B, y == A */
   x = 2*x - y;
     /* x == A - 2*B, y == A */
```

1.5
```
/* x == A, y == B */
x = 3*x +  y; y = x - y; x = x - y;
.....

   (a) /* x == B, y == A */
-->(b) /* x == B, y == 3*A */
   (c) /* x == 3*B, y == A */

     /* x == A, y == B */
     /* 3*x + y == 3*A + B, y == B */
   x = 3*x + y;
     /* x == 3*A + B, y == B */
     /* x == 3*A + B, x - y == 3*A */
   y = x - y;
     /* x == 3*A + B, y == 3*A */
     /* x - y == B, y == 3*A */
   x = x - y;
     /* x == B, y == 3*A */
```

1.6
```
/* x == A, y == B */
x = 2*x + 2*y; y = 2*x - y; x = y - x;
.....

   (a) /* 2*y - 4*x == A, y - x == B */
   (b) /* x == 4*B - 6*A, y == B - 2*A */
-->(c) /* x == 2*A + B, y == 4*A + 3*B */

     /* x == A, y == B */
     /* 2*x + 2*y == 2*A + 2*B, y == B */
   x = 2*x + 2*y;
     /* x == 2*A + 2*B, y == B */
     /* x == 2*A + 2*B, 2*x - y == 4*A + 3*B */
   y = 2*x - y;
     /* x == 2*A + 2*B, y == 4*A + 3*B */
     /* y - x == 2*A + B, y == 4*A + 3*B */
   x = y - x;
     /* x == 2*A + B, y == 4*A + 3*B */
```

Problem 2: Time complexity (20 points)

1.
```
int d = 2;
while (d*d <= N) {
  if (N%d == 0) {
    break;
  }
  d++;
}
```
(a) $O(\log N)$ **(B)** $\underline{O(\sqrt{N})}$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

2.
```
int s = 0;
for (int i = 0; i < N; i++) {
  s += i;
}
for (int i = s; i > 1; i -= 2) {
  s++;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ **(E)** $\underline{O(N^2)}$

3.
```
int d = 2, n = N;
while (n > 1) {
  if (n%d == 0) {
    n = n/d;
  } else {
    d++;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ **(C)** $\underline{O(N)}$ (d) $O(N \log N)$ (e) $O(N^2)$

4.
```
int s = 0;
for (int i = 0; i < N; i++) {
  for (int j=1; j < i; j *= 2) {
    s++;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ **(D)** $\underline{O(N \log N)}$ (e) $O(N^2)$

5.
```
int a = 42, n = N, p = 1;
while (n > 0) {
  if (n%2 == 1) {
    p *= a;
   }
   a = a*a;
   n /= 2;
}
```
(A) $\underline{O(\log N)}$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

6.
```
int i = 0, j = 0, s = 0;
while (i < N) {
  i += j;
  j++;
  for (int k=0; k*k < N; k++) {
    s++;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ **(C)** $\underline{O(N)}$ (d) $O(N \log N)$ (e) $O(N^2)$

Problem 3: Longest Subsequence (15 points)
The input of this problem is a series of lower case letter from the alphabet (i.e. 'a'..'z'). For each letter that occurs in the input, the output must show the length of the longest consecutive subsequence containing only that letter. Take for example the sequence abbaababb consisting of a's and b's. In this sequence, aa and bb are the longest consecutive subsequences of repeating characters, each having length 2. Note that a single character is a sequence that has length 1.

The input of this problems consists of a string containing lower case letters. On the output, for each occurring letter in the input, you should print the letter followed by a colon (':') and the length of the longest subsequence for that letter (see examples below). Note that the output must be in alphabetical order.

Example 1:
input:
abbaababb
output:
a:2
b:2

Example 2:
input:
aaabb
output:
a:3
b:2

Example 3:
input:
abzzccczzba
output:
a:1
b:1
c:3
z:2

```
int main(int argc, char *argv[]) {
  int longest[26];
  for (int i=0; i < 26; i++) {
    longest[i] = 0;
  }

  char c, current;
  scanf("%c", &current);
  while (current != '\n') {
    int length = 1;
    scanf("%c", &c);
    while (c == current) {
      length++;
      scanf("%c", &c);
    }
    if (length > longest[current - 'a']) {
      longest[current -'a'] = length;
    }
    current = c;
  }
  for (int i=0; i < 26; i++) {
    if (longest[i] > 0) {
      printf("%c:%d\n", 'a'+i, longest[i]);
    }
  }
  return 0;
}
```

Problem 4: Maximum Sum (15 points)
The input for this problem consists of two lines. The first line contains an integer n, where $3 \leq n \leq 35$. The next line contains n positive integers. The output should be the maximum sum of non-adjacent values from the input. Take for example the sequence `[3, 4, 6, 1, 2, 3]`. All sums without non-adjacent values are:

$$\begin{aligned} 3+6+2 &= 11 & \qquad 4+1+3 &= 8 \\ 3+6+3 &= 12 & \qquad 4+1 &= 5 \\ 3+6 &= 9 & \qquad 4+2 &= 6 \\ 3+1+3 &= 7 & \qquad 4+3 &= 7 \\ 3+1 &= 4 & \qquad 6+2 &= 8 \\ 3+2 &= 5 & \qquad 6+3 &= 9 \\ 3+3 &= 6 & \qquad 1+3 &= 4 \end{aligned}$$

The maximum sum is 12, which should be the output for this example.
Note that the sum $4 + 6 + 3 = 13$ is greater than 12, but it uses the two adjacent values 4 and 6, which is not allowed. Also note, which is clear from the above example, that in this problem a sum must have at least two terms.

Example 1:
input:
```
6
3 4 6 1 2 3
```
output:
```
12
```

Example 2:
input:
```
3
40 41 2
```
output:
```
42
```

Example 3:
input:
```
3
27 81 55
```
output:
```
82
```

```
int max(int a, int b) {
  return (a > b ? a : b);
}

int maxsum(int length, int arr[]) {
  if (length == 3) { // a sum must have at least 2 terms
    return arr[0] + arr[2];
  }
  int sumWithPrevious = arr[0];
  int sumWithoutPrevious = 0;
  for (int i = 1; i < length; i++) {
    int best = max(sumWithPrevious, sumWithoutPrevious);
    sumWithPrevious = sumWithoutPrevious + arr[i];
    sumWithoutPrevious = best;
  }
  return max(sumWithPrevious, sumWithoutPrevious);
}

int main(int argc, char *argv[]) {
  int length;
  scanf("%d", &length);
  int *arr = malloc(length*sizeof(int));
  for (int i=0; i < length ; i++) {
    scanf("%d", &arr[i]);
  }
  printf("%d\n", maxsum(length, arr));

  free(arr);
  return 0;
}
```

Problem 5: Longest Palindromic Sequence (20 points)
The input for this problem consists of two lines. The first line holds a single integer n (where $1 \leq n \leq 25$). The second line contains a string containing n lower-case letters from the alphabet (i.e. 'a'..'z').

The output should be the length of the longest palindromic subsequence that can be obtained by removing zero or more letters from the input string.

Take for example the word `characters`. Removing the characters `h`, `t`, `e`, `r` and `s` yields the longest possible palindromic subsequence `carac` which has length 5.

Example 1:
input:
```
10
characters
```
output:
```
5
```

Example 2:
input:
```
9
recursion
```
output:
```
3
```

Example 3:
input:
```
10
abbaisback
```
output:
```
5
```

```
int max(int x, int y) {
  return (x > y ? x : y);
}

int maxPalindrome(char *str, int i, int j) {
  if (i >= j) {
    return (i == j);
  }
  if (str[i] == str[j]) {
    return 2 + maxPalindrome(str, i + 1, j - 1);
  }
  return max(maxPalindrome(str, i, j - 1), maxPalindrome(str, i + 1, j));
}

int main(int argc, char* argv[]) {
  int n;
  scanf("%d", &n);

  char str[26];
  scanf("%s", str);
  printf("%d\n", maxPalindrome(str, 0, n - 1));

  return 0;
}
```