Shiv Nadar University

CSD101: Introduction to Computing and Programming

Midsem Exam

Max marks:75 9-10-2021
Time: 9.00-11.00am (Upload by 11.05am)

1. Answer all 4 questions.
2. For all program fragments assume that they are embedded in programs that compile and link without errors.
3. Please do not collaborate.
4. **Please upload by 11.05am. Submissions later than this may be penalized.**

1. (a) Consider the `while` loop below where `<condition>` is the condition in the while loop and `<statement>` is a simple or compound statement that forms the loop-body.

```
while (<condition>) {
    <statement>
}
```

Write two code fragments that use a) just the `for` loop and b) just the `do-while` loop that behave exactly like the given `while` loop.

Solution:

a)
```
//inital and increment are blank
for ( ;<condition>; ) {
    <statement>
}
```

b)
```
if (<condition>)
    do {
        <statement>
    } while <condition>;
```

No marks if any mistake in the fragments.

(b) Consider the following code fragment that has a nested, compound `if-then-else` statement. Assume the function `charType(c)` is already defined and `c` has been initialized to a character.

```
int type=charType(c); //c is a char
if (type==0) printf("Digit\n");
else
   if (type=1) printf("UC Alphabet\n");
```

```
   else
      if (type==2) printf("LC Alphabet\n");
      else
         if (type==3) printf("Punctuation\n");
         else printf("Ctrl/non-printable\n");
exit(0);
```

Rewrite the above code fragment to use `switch` statement(s) (without any `if-then-else`) so that it works exactly like the code fragment above.

Solution:

```
switch (charType(c)) {
    case 0: {printf("Digit\n"); break;}
    case 1: {printf("UC Alphabet\n"); break;}
    case 2: {printf("LC Alphabet\n"); break;}
    case 3: {printf("Punctuation\n"); break;}
    default: {printf("Ctrl/non-printable\n");}
}
exit(0);
```

Another possibility:

```
int type=charType(c); //c is a char
switch (type) {
    <same as above>
}
```

1 mark less for each missing `break` or missing `case`.

[(3,4),6,=13]

2. (a) Suppose you had 16-bits to represent integer data. Then answer the following (give your values in decimal numbers or as expressions):

   i. What is the largest unsigned integer that can be represented?

   Solution:
   $2^{16} - 1 = 65535$

   ii. If one bit is reserved for the sign then what are the smallest and largest integers that can be represented?

   Solution:
   smallest$=-2^{15} = -32768$, largest$=2^{15} - 1 = 32767$

   (b) The code fragment shown below converts n to its binary representation. So, for example, if $n = 12$ then it will convert it to `1100`.

   Fill in the missing parts indicated by `?1?` to `?4?` appropriately so that the code works correctly.

```
/*Converts n to its binary representation.*/
int bin=0, tens=1, n, tmp;
printf("Give n = ");
scanf("%d",&n);
tmp=n;
while (?1?) {
   bin=?2?;
   tens=?3?;
   n=?4?;
}
printf("Binary representation of %d is %d\n", tmp, bin);
exit(0);
```

Solution:

```
?1? n>0
?2? bin+tens*(n%2)
?3? tens*10
?4? n/2
```

(c) What will be the output of the program fragment in (b) if $n = 27$?

Solution:

Binary representation of 27 is 11011

[(1,2,2),(2,3,2,2),3=17]

3. Recall (from Quiz 1) that the prime factorization of a positive integer n yields a sequence of prime factors which when multiplied together give n. For example, if $n = 60$ then the sequence is: $2, 2, 3, 5$. Assume function `factorize(int factors[ ], int n)` prime factorizes the integer n and stores the factors in the array `factors` starting at index 1. The length of the sequence is stored at index 0. If $n = 60$ then `factors` will contain the sequence: `4, 2, 2, 3, 5` stored from indices 0 to 4.

(a) Define a function: `void printFactors(int f[])` that takes as input the factors of a positive integer n created by `factorize(int factors[], int n)` and prints out the sequence of factors separated by a space with a new line at the end of the sequence.

Solution:

```
void printFactors(int f[]) {
   for (int i=1; i<=f[0]; i++) printf("%d ",f[i]);
   printf("\n");
   return;
}
```

One mark less if you don't print the space after %d. One mark less if you print n after %d or don't print the final n. One mark less if you start the for loop from 0 instead of 1. One mark less if the termination condition is `<f[0]` instead of `<=f[0]`.

(b) The code fragment given below finds the greatest common divisor (GCD) of two integers m, n by using the function `factorize` to prime factorize each integer. Fill in the missing parts labelled `?1?` to `?6?` suitably so that the program works correctly. Assume that the functions `factorize` and `printFactors` have been defined and are available.

```
int m, n, mi=1, ni=1, fm[50], fn[50], gcd=1;
/*m, n - the two integers whose gcd is to be calculated.
fm, fn - arrays storing result of factorizing m, n resp.
mi, ni - indicies into fm, fn pointing to the first factor resp.
gcd - stores the gcd of m, n.
*/
scanf("%d%d",&m, &n);
factorize(m, fm);
printFactors(fm);
factorize(n,fn);
printFactors(fn);
while (?1?) {
   if (fm[mi]==fn[ni]) {?2?; ?3?; ?4?;}
   else
      if (fm[mi]<fn[ni]) ?5?;
      else ?6?;
}
printf("GCD of %d and %d is %d.\n", m,n,gcd);
exit(0);
```

Solution:

`?1?` `mi<=fm[0] && ni<=fn[0]`
`?2?` `gcd*=fm[mi]` or `gcd*=fn[ni]` or equivalent
`?3?` `mi++` or equivalent
`?4?` `ni++` or equivalent
`?5?` `mi++` or equivalent
`?6?` `ni++` or equivalent

Ordering of `?2?`, `?3?`, `?4?` should be such that the index used in calculating `gcd` should not change before the `gcd` calculation. So, for example `?3?` and `?4?` can be interchanged.

(c) What will be printed when the code in part (b) above is executed assuming the values read in are: $m = 128$ and $n = 60$.

Solution:

```
2 2 2 2 2 2 2
2 2 3 5
GCD of 128 and 60 is 4.
```

One mark less if the number of factors is also printed at the beginning. That is `f[0]` is also printed.

[5,(4,2,2,2,2,2),6=25]

4. (a) Recall that in C characters are encoded as unsigned integers. The program fragment below reads upto 30 printable characters and changes the case of every alphabetic character (i.e. A-Z, a-z) from upper case to lower case and vice versa. Remember the codes of A-Z (and a-z) are sequential. Some parts of the program are replaced by `?1?` to `?8?`. Fill in code for the replaced parts so that the program works correctly.

```
/*Reads a string of upto 30 printable characters and changes
the case of every alphabetic character from upper case to
lower case and vice versa.
*/
char str[?1?], c;
int i=0;
printf("Give the string (<=30 chars)=");
while ((c=getchar())!=?2? && i<30) str[?3?]=c;
str[?4?]='\0';
for (i=0; (c=str[i])!=?5?; i++)
   if (c>='A' && c<='Z') str[i]+=?6?;
   else
      if (?7?) str[i]+=?8?;
printf("%s\n",str);
exit(0);
```

Solution:

```
?1? 31
?2? '\n'
?3? i++
?4? i
?5? '\0'
?6? ('a'-'A') or equivalent
?7? (c>='a' && c<='z') or equivalent
?8? ('A'-'a') or equivalent
```

(b) What will be printed out by the code fragment in part (a) if the input is:
`"IsThisAStriNGOf25Chars?"`?

Solution:

`"iStHISasTRIngoF25cHARS?"`

[(2×8),4=20]