Exercise 1: Simple Programs

PRAVEEN KUMAR R

30/01/2018

Assignment	1
Reg No	312217104114
Name	PRAVEEN KUMAR R
Grade	
Date	

1 Home work

Objective

To gain some familiarity with variables, assignment, output statement, control flow, functions, and arrays in C.

Output (printf) statements

Write a program to print this text, with the second and the fourth lines indended, on the stdout.

```
The heights by great men reached and kept
   Were not attained by sudden flight,
But they, while their companions slept,
   Were toiling upward in the night.
```

1. Create a new program file `longfellow.c` in emacs. Edit your program.
2. Create a makefile to compile your source program file `longfellow.c`, compile it to an executable program `longfellow`.
3. If there are any errors as you compile, fix them.
4. List the errors which occurred.

Program

```
#include<stdio.h>
int main()
{
  char a[100];
  int i=0;
  while(fgets(a,100,stdin)!=NULL)
    {
      if(i%2==1)
printf("\t");
      printf("%s",a);
      i++;
    }

  return 0;
}
```

Test

- **Input**

```
The heights by great men reached and kept
Were not attained by sudden flight,
But they, while their companions slept,
Were toiling upward in the night.
```

- **Output**

```
        The   heights  by       great     men         reached  and      kept
              Were     not      attained  by          sudden   flight,
        But   they,    while    their     companions  slept,
              Were     toiling  upward    in          the      night.
```

Minimum of three numbers

Write a program `min2.c` to read two numbers from stdin and print the smallest of the two numbers.

1. Implement the functionality in `main()`.

2. Divide your program into `main()` and another function `min2()`. Function `min2()` takes two numbers as inputs and returns as output the minimum of the two inputs.

3. Design a function `min3()` that takes three numbers as inputs and returns as output the minimum of the three inputs.

4. Design `min3()` in at least three different ways. Make one comment, good or bad, about each of the designs.

Program

```
#include<stdio.h>
int min2(int a,int b)
{
  return a>b?b:a;
}
int min3(int a,int b,int c)
{
  return a<min2(b,c)?a:min2(b,c);
}
int main()
{
  int a,b,c;
  scanf("%d%d%d",&a,&b,&c);
  printf("%d",min3(a,b,c));
  return 0;
}
```

Test

- **Input**

```
34 56 76
```

- **Output**

```
34
```

Power x^n

Construct a program `power.c`.

1. A number b raised to the power m, b^m, can be calculated by cumulatively multiplying 1 by b, m times. For example, if $b = 2$ and $m = 5$, then the process for calculating $b^m = 2^5$ proceeds as shown in the table below:

p	p * 2
1	
2^1	1 * 2
2^2	2^1 * 2
2^3	2^2 * 2
2^4	2^3 * 2
2^5	2^4 * 2

   Implement a program to calculate the power b^m. Hardcode b and m into your program (no need to read them from the user). Print the output, for example, as `2^5 = 32`.

Program

```
#include<stdio.h>
int power(int a,int n);
int power(int a,int n)
{
  if(n==1)
    return a;
  if(n%2==0)
    {
      int p=power(a,n/2);
      return p*p;
    }
  else
    return a*power(a,n-1);
}
int main()
{
  int a=7,n=6;
  printf("%d^%d=%d",a,n,power(a,n));
  return 0;
}
```

Test

7^6=117649

1. Read b and m from the user (stdin) for the power program. First, print a prompt message to the user:

   ```
   Enter the base and the exponent:
   ```

2. Define a function `power(x, n)` that raises x to the power n. It takes `x` and `n` as parameters and returns the power x^n as the result. Write the code for `power()` before `main()`. Call `power()` from `main()`.
3. Write the code for `power()` *after* `main()` and see the errors reported. Fix it: Let the code for `power()` be after `main()`. But write the prototype of `power()` before `main()`.
4. In `power(x, n)`, instead of using a variable to count the number of iterations completed, use the parameter `n` to count the number iterations left to terminate the loop.

Program

```
#include<stdio.h>
 int power(int a,int n);
 int power(int a,int n)
 {
   if(n==1)
     return a;
   if(n%2==0)
```

```
    {
      int p=power(a,n/2);
      return p*p;
    }
  else
    return a*power(a,n-1);
}
int main()
{
  int a,n;
  scanf("%d%d",&a,&n);
  printf("%d^%d=%d",a,n,power(a,n));
  return 0;
}
```

Test

$$4^6=4096$$

1. List four idioms for repeating a loop n times.

Example 1

```
int i=0;]
while(i<n)
   i++;
```

Example 2

```
for(int i=0;i<n;i++);
```

Example 3

```
int i=n-1;
while(i>=0)
   i--;
```

Example 4

```
for(int i=n-1;i>=0;i--);
```

1. Pass a negative exponent to `power()`. What is the error that occurs in the run time? Rename `power()` as `pos_power()` and write a function `power()` that works correctly for any integer exponent, positive or negative.

Program

```
#include<stdio.h>
 float power(float a,float n);
 float power(float a,float n)
 {
   if(n==1)
     return a;
   if((int)n%2==0)
     {
       int p=power(a,n/2);
       return p*p;
     }
   else
     return a*power(a,n-1);
 }
 int main()
 {
   float a,n,p;
   scanf("%f%f",&a,&n);
    p=power(a,n<0?-n:n);
   if(n<0)
     p=1/p;
   printf("%0.0f^%0.0f=%f",a,n,p);
   return 0;
 }
```

Test

3^{-5}=0.004115

Table of powers

Populate a table with powers b^m for a given range of m.

1. Modify your program `main()` to print powers b^m for a number b for powers from 0 to 20. Read `b` and `m` from the user. Format the output as shown below.

```
2^0  =       1
2^1  =       2
2^2  =       4
2^3  =       8
2^4  =      16
...
2^20 = 1048576
```

Program

```
#include<stdio.h>
int main()
{
  long int a,n,p=1;
  scanf("%ld%ld",&a,&n);
  for(int i=0;i<=n;i++)
      {
printf("%ld^%d=%ld\n",a,i,p);
p=p*a;
      }
}
```

Test

- **Input**

```
3 15
```

- **Output**

3^{0}=1
3^{1}=3
3^{2}=9
3^{3}=27
3^{4}=81
3^{5}=243
3^{6}=729
3^{7}=2187
3^{8}=6561
3^{9}=19683
3^{10}=59049
3^{11}=177147
3^{12}=531441
3^{13}=1594323
3^{14}=4782969
3^{15}=14348907

1. Declare an array of 100 numbers. Store 10 numbers in the array. Write a function `print_array()` to print a subarray. The function takes three parameters, the array name `a`, the lower bound `low`, and the upper bound `high` of the subarray. Remember that the upper bound is open — the subarray `a[low:high]` consists of `a[low] ... a[high-1]` (`a[high]` is not a part of the subarray). Drive `print_array()` from `main()`.

Program

```
#include<stdio.h>
void print_array(int a[], int low, int high);
void print_array(int a[], int low, int high)
{
  printf("{");
  for(int i=low;i<high;i++)
    printf("%d %c ",a[i],i<high-1?’,’:’ ’);
  printf("}");
}
int main()
{
  int a[100]={0,2,4,6,8,10,12,14,16,18};
  print_array(a,2,8);
}
```

Test

```
{4   6   8   10   12   14 }
```

1. Fill the subarray `a[0]...a[20]` with 2^0 to 2^{20}. Print it.

Program

```
#include<stdio.h>
 int main()
 {
   long int a=2,n=20,p=1,m[100];
   for(int i=0;i<=n;i++)
     {
       m[i]=p;
       p=p*a;
     }
   for(int i=0;i<=n;i++)
       {
 printf("%d\n",m[i]);
       }
 }
```

1. Fill the subarray `a[0]...a[20]` with 2^{-10} to 2^{10}. Print it.

```
#include<stdio.h>
 float power(float a,int n);
 float power(float a,int n)
 {
   if(n==0)
     return 1;
   if(n%2==0)
     {
      float p=power(a,n/2);
       return p*p;
     }
   else
     return a*power(a,n-1);
 }
 int main()
 {
   int k=0;
   float a=2,n=10,p,m[21];
   for(int i=-10;i<=n;i++)
     {
       p=power(a,(i<0?-i:i));
       if(i<0)
p=1/p;
       m[k++]=p;
     }
   for(int i=0;i<k;i++)
     printf("%f \n",m[i]);
   return 0;
 }
```

Test

```
0.000977
0.001953
0.003906
0.007812
0.015625
 0.03125
  0.0625
   0.125
    0.25
     0.5
     1.0
     2.0
     4.0
     8.0
    16.0
    32.0
    64.0
   128.0
   256.0
   512.0
  1024.0
```

2 Questions

Answer the following questions.

1. How many lines are printed by `printf("Hello,\nworld!\nBye,\nworld!")`?

   (a) 1

   (b) 2

   (c) 3

   (d) 4

   Answer:4

2. What is the output?

   ```
   a = 5; b = 10;
   a = b;
   b = a;
   printf ("a = %d, b = %d\n", a, b);
   ```

   Answer: a=10, b=10

3. What is the output? What does the code do?

   ```
   a = 5; b = 10;
   a = a+b;
   b = a-b;
   a = a-b;
   printf ("a = %d, b = %d\n", a, b);
   ```

   Answer: a = 10, b = 5

4. What is the output? What does the code do?

   ```
   a = 5; b = 10; c = 15;
   t = a;
   a = b;
   b = c;
   c = t;
   printf ("a = %d, b = %d, c = %d\n", a, b, c);
   ```

   Answer: a=10, b=15, c=5

5. Translate the expression $d = \sqrt{b^2 - 4ac}$ into C statement.

   Answer: d= sqrt(b*b-4*a*c);

6. Translate the expression $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ into C statement.

   Answer: d= sqrt(pow(x_1-x_2,2)+pow((y_1-y_2),2));

7. What is the output?

```
a = 5; b = 10;
m = a;
if (b < m)
   m = b;
printf ("%d\n", m);
```

   Answer: 5

8. What is the output?

```
mark = 40;
if (mark < 50)
  grade = ’E’;
if (mark < 60)
  grade = ’D’;
if (mark < 70)
  grade = ’C’
printf ("%c\n", grade);
```

   Answer: C

9. Trace the process generated by the loop

```
n = 5;
f = 1; i = 0;
while (i < n) {
  f = f * i;
  i = i + 1;
}
```

   Answer:

i	f	i+1	f*i

10. Write a loop (while statement) which will generate the process shown in the table.

q	r	r - 5	q + 1
0	22		
1	17	22 - 5	0 + 1
2	12	17 - 5	1 + 1
3	7	12 - 5	2 + 1
4	2	7 - 2	3 + 1

Answer:

```
r = 22;
q = 0;
while(r-5>0)
 {
    r=r-5;
    q=q+1;
 }
```