Home Work #1

Student Name: Yudhistara sai kumar Maineedi

Student ID: 0988780

Student email: ymaineedi@uh.edu

Submission Date:10/11/2011

Code:

```
#include <iostream>
#include<time.h>
#include <math.h>
#define N 10000
using namespace std;

int ARRAY[N];   /*This Array is used to compute the factorial by storing each
digit of a number in the array  */
int STOREDVALUES[N][N];/*This array is used to store the pre computed
values*/

class Fact
{
public:
void factorial(int _previousvalue,int _value) /*This function is used to
caliculate the factorial*/
{
int result=0,reminder=0;
int k;
int i=N;
for(int j=0;j<=N;j++) // getting the factorial value of the largest value
given before so that we can continue from there
ARRAY[j]=STOREDVALUES[_previousvalue][j];
for(k=_previousvalue+1;k<=_value;k++)
{
      reminder=0;

while(i>0)                          //storing the products into array
      {
      result=ARRAY[i]*k+reminder;
      if(result>9)
      {
            ARRAY[i]=result%10;
      reminder=result/10;
      }
      else
      {
ARRAY[i]=result;
      //cout<<ARRAY[i]<<endl;
      reminder=0;
      }
      i--;

      }
      i=N;
      result=0;

for(int j=0;j<=N;j++)            // storing the factorials which are computed
into STOREDVALUES
STOREDVALUES[k][j]=ARRAY[j];
}
}

void print(int _value)/*Is used to print the values which are stored in the
array while computing*/
```

```
{
      int j=0;
      for( j=0;j<=N;j++)
      {
      if(STOREDVALUES[_value][j]!=0)
            break;
      }
      cout<< "factorial of the number is:"<<endl;
    for(j;j<=N;j++)
      cout<<STOREDVALUES[_value][j];
      cout<<endl;
}
};



int main()  /*This is the main function*/
{

      Fact f;
      int* ARRAY=(int*)malloc(10000*sizeof(int));
int i,value,result=0,reminder=0;
int previousvalue=1;
for(i=0;i<N;i++)  /*This is used to intialise the values of the arrays*/
{
ARRAY[i]=0;
STOREDVALUES[1][i]=0;
}
STOREDVALUES[1][N]=1;
cout<<"Enter the value"<<endl;
cin>>value;
while(value>=0) //Loop to read n no of values untill you give a negative
number
      {
            clock_t start=clock();
            if(value>previousvalue)  // Checking whether the value entered is
greater than the largest among the values which were entered before
            {
            f.factorial(previousvalue,value);//calling the factorial function
            previousvalue=value;
            }
f.print(value);
clock_t end=clock();

cout<<"time taken:"<<end-start<<endl;
cout<<"Enter a positive value (Negative value to exit)";

cin>>value;
}
}
```

Design:

In my design I took two arrays one ARRAY for performing the computations and the other STOREDVALUES for storing the computed values of factorial(first row has 1!,2nd row has 2! ,…..). When we give a positive value as an input to the program

That value is put into ARRAY from the back such that each digit of the number occupies the each ARRAY location from the back.

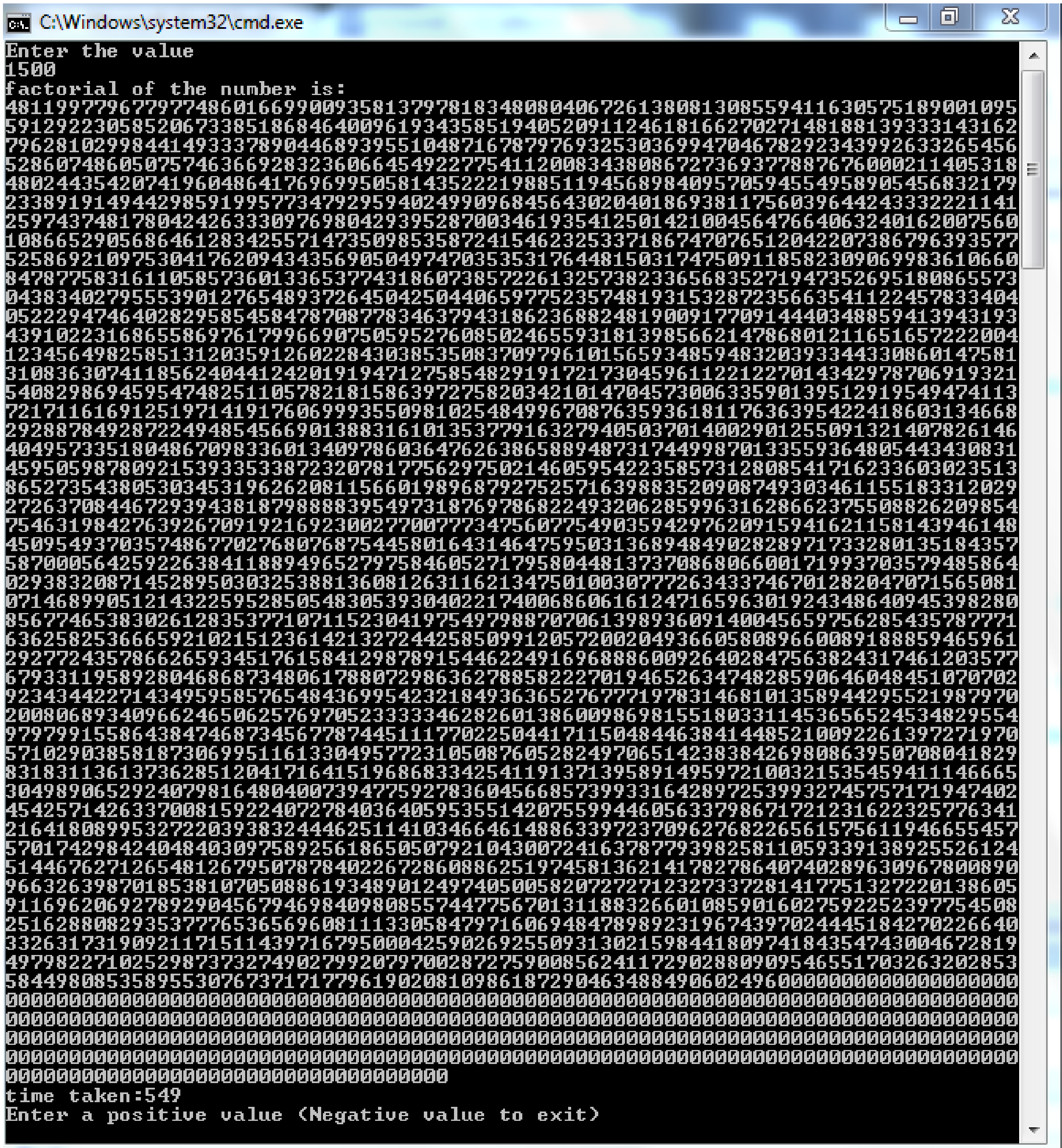
(In my factorial function i am storing all the factorials of all the values lesser than the “value” in the STOREDVALUES which are used in calculating the factorial of the “value” so that we can use it in future.)

Now I have a previousvalue variable which has the largest value for which we have computed the factorial before.

If previousvalue is lesser than value i am sending the previousvalue and value to the factorial function as we already have the factorials for the values till previousvalue in STOREDVALUES we need not do it again.we can just pick up the factorial of the previousvalue from STOREDVALUES and continue it tillvalue

If previous value is greater than the present value we need not perform the factorial (need not call factorial) as it is already stored in STOREDVALUES we can just retrieve it from the array.

Screenshot:


C:\Windows\system32\cmd.exe
Enter the value
1500
factorial of the number is:
48119977967797748601669900935813797818348080406726138081308559411630575189001095
59129223058520673385186846400961934358519405209112461816627027148188139333143162
79628102998441493337890446893955104871678797693253036994704678292343992633265456
52860748605075746366928323606645492277541120083438086727369377887676000211405318
48024435420741960486417696995058143522219885119456898409570594554958905456832179
23389191494429859199577347929594024990968456430204018693811756039644243332221141
25974374817804242633309769804293952870034619354125014210045647664063240162007560
10866529056864612834255714735098535872415462325337186747076512042207386796393577
52586921097530417620943435690504974703535317644815031747509118582309069983610660
84787758316110585736013365377431860738572261325738233656835271947352695180865573
04383402795553901276548937264504250440659775235748193153287235663541122457833404
05222947464028295854584787087783463794318623688248190091770914440348859413943193
43910223168655869761799669075059527608502465593181398566214786801211651657222004
12345649825851312035912602284303853508370977961015659348594832039334433086014758‍1
31083630741185624044124201919471275854829191721730459611221227014342978706919321
54082986945954748251105782181586397275820342101470457300633590139512919549474113
72171161691251971419176069993550981025484996708763593618117636395422418603134668
29288784928722494854566901388316101353779163279405037014002901255091321407826146
40495733518048670983360134097860364762638658894873174499870133559364805443430831
45950598780921539335338723207817756297502146059542235857312808541716233603023513
86527354380530345319626208115660198968792752571639883520908749303461155183312029
27263708446729394381879888839549731876978682249320628599631628662375508826209854
75463198427639267091921692300277007773475607754903594297620915941621158143946148
45095493703574867702768076875445801643146475950313689484902828971733280135184357
58700056425922638411889496527975846052717958044813737086806600171999370357948586‍4
02938320871452895030325388136081263116213475010030777263433746701282047071565081
07146899051214322595285054830539304022174006860616124716596301924348640945398280
85677465383026128353771071152304197549798870706139893609140045659756285435787771
63625825366659210215123614213272442585099120572002049366058089660089188859465961
29277243578662659345176158412987891544622491696888600926402847563824317461203577
67933119589280468687348061788072986362788582227019465263474828590646048451070702
92343442271434959585765484369954232184936365276777197831468101358944295521987970
20080689340966246506257697052333334628260138600986981551803311453656524534829554
97979915586438474687345677874451117702250441711504844638414485210092261397271970
57102903858187306995116133049577231050876052824970651423838426980863950708041829
83183113613736285120417164151968683342541191371395891495972100321535459411146665
30498906529240798164804007394775927836045668573993316428972539932745757171947402
45425714263370081592240727840364059535514207559944605633798671721231622325776341
21641808995327220393832444625114103466461488633972370962768226561575611946655457
57017429842404840309758925618650507921043007241637877939825811059339138925526124
51446762712654812679507878402267286088625197458136214178278640740289630967800890
96632639870185381070508861934890124974050058207272712327337281417751327220138605
91169620692789290456794698409808557447756701311883266010859016027592252397754508
25162880829353777653656960811133058479716069484789892319674397024445184270226640
33263173190921171511439716795000425902692550931302159844180974184354743004672819
49798227102529873732749027992079700287275900856241172902880909546551703263202853
58449808535895530767371717796190208109861872904634884906024960000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000
time taken:549
Enter a positive value (Negative value to exit)