Templates

Templates

```
//theSmallest template
template <class T> T theSmallest(T *V, int size)
{
  T temp=V[0];
  for(int i=1; i<size; i++)
    if(temp>V[i]) temp = V[i];
  return temp;
}

#ifndef __CIRCLE_H_
#define __CIRCLE_H_

class Circle
{
  float m_xCoordinate;
  float m_yCoordinate;
  float m_radius; //radius
 public:
  Circle();
  bool  operator > (const Circle &i_circle); //overloading > operator
  void printCircle();
};
#endif
```

Circle.cpp

```
//initialize the radius at rondom
Circle::Circle()
{
  m_xCoordinate = (int)(5.0*rand()/(RAND_MAX));;
  m_yCoordinate = (int)(5.0*rand()/(RAND_MAX));;
  m_radius =10.0*rand()/(RAND_MAX);
}

//overloading the > operator
bool Circle::operator > (const  Circle &i_circle)
{
 return (m_radius > i_circle.m_radius);
}

void Circle::printCircle()
{
  std::cout<<"Centre = ("<<m_xCoordinate<<","<<m_yCoordinate<<")   ";
  std::cout<<"Radius = "<<m_radius<<std::endl<<std::endl;
}
```

Main.cpp

```
int main()
{
    int a[]  = {10,20,3,4,5,56,43,1,21,54};
    int L = theSmallest(a,10);
    std::cout << L << std::endl;

    //template test on an array of user defined objects: Circles
    Circle CC[10];
    for(int i=0; i<10; i++)
    CC[i].printCircle();

    Circle S;
    S=theSmallest(CC,10);
    std::cout<<"The␣smallest␣circle␣is␣"<<std::endl;
    S.printCircle();

    return 1;
}
```