1. (a) (4) Describe the output of this programme.

```
#include<cstdlib>
#include<iostream>

using namespace std;

int main()
{
      cout<<"Of man's first disobedience\n";
      cout<<"and the fruit of that forbidden tree."<<endl;
}
```

*Solution:*This program outputs

```
Of man's first disobedience
and the fruit of that forbidden tree.
```

(b) (4) What will this programme output

```
#include<cstdlib>
#include<iostream>

using namespace std;

int main()
{
      int a=0;

      cout<<a<<endl;

      a+=10;

      cout<<a<<endl;

      a=a%4;

      cout<<a<<endl;

}
```


*Solution:*This program outputs

```
0
10
2
```

The '2' is 10 modular 4.

(c) (4) Explain why this programme outputs what it does.

```
#include<cstdlib>
#include<iostream>

using namespace std;

int main()
{
      unsigned int a=1;
      unsigned int b=2;

      cout<<b-a<<endl;
      cout<<a-b<<endl;
}
```

*Solution:*This outputs

```
1
a large number
```

where the large number is the largest number stored by an int. If `a` and `b` are `unsigned int`s then `a-b` will be too, since this should be -1, and an `unsigned int` can't be negative you get the large number.

(d) (4) I ran this programme

```
#include<cstdlib>
#include<iostream>

using namespace std;

int main()
{
```

```
   cout<<rand()%10<<endl;

}
```

and got the output '3'. What would I get if I ran it again and why?

Solution:'3' again. The random number generator is deterministic, it produces a new random number from the previous one using an exact algorithm, but one that is designed to produce output with the statistical properties of random numbers. When run run the programme it starts from the same place and so gives the same output. To change this you need to give it a different starting point, this is done using the function `srand(int seed)`.

(e) (4) Write a short programme that outputs a random number in $[0, 1]$.

Solution:

```
#include<cstdlib>
#include<iostream>

using namespace std;

int main()
{
   cout<<double(rand())/double(RAND_MAX)<<endl;

}
```

`RAND_MAX` gives the largest number `rand()` can produce.

2. (a) (4) Describe the output of this programme

```
#include<cstdlib>
#include<iostream>

using namespace std;

int main()
{
```



```
      for(int i=0;i<10;i++)
            cout<<i<<endl;

}
```

*Solution:*So the for loop goes around as `i` advances from 0 to 9:

```
0
1
2
3
4
5
6
7
8
9
```

(b) (3) Describe the output of this programme

```
#include<cstdlib>
#include<iostream>

using namespace std;

int main()
{
      for(int i=0;i<10;++i)
            cout<<i++<<endl;

}
```

*Solution:*Every time it goes around the loop `i` is increased by one after it has been printed out, because of the plus plus and is then increased again by the for so

02468 (1)

(c) (5) Describe the output of this programme

```
#include<cstdlib>
```

```
#include<iostream>

using namespace std;

int main()
{
      int i=2;
      for(int i=0;i<3;i++)
            cout<<i<<endl;
      cout<<i<<endl;
}
```

*Solution:*The `i` in the for loop is local to it, so we get

```
0
1
2
2
```

where the last 2 is the other `i`.

(d) (8) What will the output of this program be.

```
#include<cstdlib>
#include<iostream>

using namespace std;

int main()
{
      a=18;
      b=12;

      c=b;

      while(a%c!=0&&b%c!=0)
            c--;

      cout<<c<<endl;

}
```


How would it need to be changed to output the highest common factor of `a` and `b`.

*Solution:*The programme will not produce an output since the three `int` variables are not declared. If this is fixed it outputs '12', to output the highest common factor, the `&&` needs to be changed to `||`; both divisions have to have zero remained for `c` to be the highest common factor.

3. (a) (2) Describe the output of this programme

```
#include<cstdlib>
#include<iostream>

using namespace std;

void print(double a)
{
      cout<<"bam "<<a<<endl;
}

int main()
{
      double a=6.5;
      print(a);
}
```

Solution:

```
bam 6.5
```

(b) (2) Describe the output of this programme

```
#include<cstdlib>
#include<iostream>

using namespace std;

void print(int a)
{
```

```
      cout<<"bam "<<a<<endl;
}

int main()
{
      double a=6.5;
      print(a);
}
```

*Solution:*This time the `double` gets cast to an `int`, so,

```
bam 6
```

(c) (2) Describe the output of this programme.

```
#include<cstdlib>
#include<iostream>

using namespace std;

void print(int a)
{
      cout<<"bam "<<a<<endl;
}

void print(double a)
{
      cout<<"bim "<<a<<endl;
}


int main()
{
      double a=6.5;
      print(a);
}
```

*Solution:*Now the `double` version of `print` is called, hence


```
bim 6.5
```

(d) (2) Describe the output of this programme.

```
#include<cstdlib>
#include<iostream>

using namespace std;

void print(int a)
{
      cout<<"bam "<<a<<endl;
}

void print(double a)
{
      cout<<"bim "<<a<<endl;
}


int main()
{
      double a=6.5;
      print(int(a));
}
```

*Solution:*And this time the `int` version

```
bam 6
```

(e) (2) This programme will not compile, can you say why?

```
#include<cstdlib>
#include<iostream>

using namespace std;

void print(int a)
{
      cout<<"bam "<<a<<endl;
```

```
}

void print(double a)
{
      cout<<"bim "<<a<<endl;
}


int main()
{
      double a=6.5;
      print((unsigned int)(a));
}
```

*Solution:*The call to `print` is ambigious, it doesn't know if you want the `double` or the `int` version.

(f) (2) Describe the output of this programme.

```
#include<cstdlib>
#include<iostream>

using namespace std;

void print(int a)
{
      cout<<"bam "<<a<<endl;
}

void print(double a)
{
      cout<<"bim "<<a<<endl;
}


int main()
{
      print(2);
}
```


Solution:'2' is assume to be an `int`, so

```
bam 2
```

(g) (4) What does the `const` do in the argument of a function declaration and explain why you might use it.

*Solution:*If the value of a `const` variable is changed in the function it returns an error. You would use a declaration like this if you are passing by reference in order to save the space and time required to do a copy but in a situation where you don't want the function to change the value.

(h) (4) This programme won't compile. Correct the mistake and describe the output.

```
#include<cstdlib>
#include<iostream>

using namespace std;

void add(int a,const int & b);

int main()
{
      int a=4;
      int b=-2;

      add(a,b);

      cout<<a<<" "<<b<<endl;

}

void add(int b,const int &a)
{
      a+=b;
      b+=a;
}
```

*Solution:*It won't compile because a isn't constant, you could fix it by removing the line
tt a+=b in which case the output is `4 -2` or you could remove the `const` from the delaration and the function itself, in which case the output would be `2 0`.

4. (a) (6) Describe the output of this programme

```
#include<cstdlib>
#include<iostream>
#include<vector>

using namespace std;

int main()
{

      vector<vector<double> > m;

      {
        vector<double> line1, line2;
        line1.push_back(0);
        line1.push_back(1);
        line2.push_back(2);
        line2.push_back(3);
        m.push_back(line1);
        m.push_back(line2);
      }

      cout<<m.at(0).at(1)<<" "<<m.at(1).at(0);
      cout<<" "<<m.at(0).at(0)<<endl;

      {
        vector<double> line1, line2;
        line1.push_back(0);
        line1.push_back(1);
        m.push_back(line1);
      }
```



```
      cout<<m.at(0).at(1)<<" "<<m.at(1).at(0);
      cout<<" "<<m.at(0).at(0)<<endl;
      cout<<m.at(2).at(1)<<" "<<m.at(1).at(1)<<endl;

      cout<<m.size()<<endl;
      cout<<m.at(0).size()<<endl;

}
```

```
1 2 0
1 2 0
1 3
3
2
```

(b) (6) What is a `reference`, a `pointer` and an `iterator`.
*Solution:*A `reference` is an alias for a variable, what is done to the reference is done to the variable. A pointer stores the memory location of a variable, the dereferencing operator can be used to give the value stored in that location. An iterator is a pointer to an element in a container class, it has an increment operator which moves forward by one the element it is pointing to.

(c) (4) What is the difference between a `list` and a `vector`?
*Solution:*A `list` has no index and doesn't allow direct access to the elements, however, new elements can be added anywhere on a list in finite time.

(d) (4) Describe the output of this programme.

```
#include<cstdlib>
#include<iostream>
#include<list>
#include<string>

using namespace std;

int main()
```

```
{

	list<string> words;
	int n=2;

	words.push_back("lets");
	words.push_back("go");
	words.push_back("completely");
	words.push_back("crazy");

	list<string>::iterator it=words.begin();

	for(int i=0;i<n;i++)
	{
		cout<<*it<<" ";
		it++;
	}

	cout<<endl;

}
```

*Solution:*This outputs `lets go`, we go around the `for` loop twice and each time the `interator` is dereferenced in the `cout` and is incremented.