Question 1

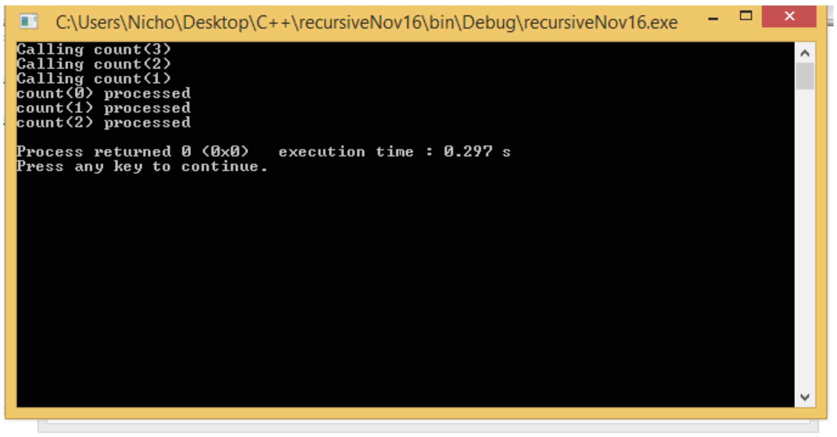
1.1 strcpy(saveWord, word); // C-strings use the function strcpy to copy one string to another

1.2 if(strcmp(word, saveWord)) // C-strings use the function strcmp to compare two strings

1.3 int j = 4;

Question 2

2.1


C:\Users\Nicho\Desktop\C++\recursiveNov16\bin\Debug\recursiveNov16.exe

```
Calling count(3)
Calling count(2)
Calling count(1)
count(0) processed
count(1) processed
count(2) processed

Process returned 0 (0x0)   execution time : 0.297 s
Press any key to continue.
```

2.2

void count(int counter)

2.3

count(--counter);

2.4

assert(counter>=0);

Question 3

3.1

a = (*p1 *2) /5;

3.2

Creating a new int nameless variable an assigning the pointer p1 to point to it

3.3

We cannot assign the address of pointer p1 to another pointer p2, because it has been deleted

3.4

```
p3 = &a;
```

3.5

To access actual value that pointer p3 is pointing to.

Question 4

4.1

```
#ifndef BURSARY_H
#define BURSARY_H

#include <iostream>
using namespace std;

class Bursary
{
  public:
    Bursary();
    Bursary(int studentNumberP, int yearsOfStudentP, int modulesPassedP);
    ~Bursary();
    int get_Number();
    void yearOfStudy(int yearsOfStudent);
    friend bool operator==(const Bursary & bursary1, const Bursary & bursary2);
    friend istream& operator>>(istream& ins, Bursary& b);

  private:
    int studentNumber;
    int yearsOfStudent;
    int modulesPassed;

};
```

```
#endif // BURSARY_H
```

4.2

```
#include <iostream>
#include "Bursary.h"

using namespace std;

Bursary::Bursary()
{
    studentNumber = 0;
    yearsOfStudent = 0;
    modulesPassed = 0;
}

Bursary::Bursary(int studentNumberP, int yearsOfStudentP, int modulesPassedP)
{
    studentNumber = studentNumberP;
    yearsOfStudent = yearsOfStudentP;
   modulesPassed = modulesPassedP;
}

Bursary::~Bursary(){}

int Bursary::get_Number()
{
    return studentNumber;
}


void Bursary::yearOfStudy(int yearsOfStudentP)

{

   yearsOfStudent = yearsOfStudentP;

}


bool operator==(const Bursary & bursary1, const Bursary & bursary2)

{

   if(bursary1.yearsOfStudent == bursary2.yearsOfStudent && bursary1.modulesPassed ==
bursary2.modulesPassed)

      return true;

   else

       return false;

}


istream& operator>>(istream& ins, Bursary & b)

{

   ins >> b.modulesPassed >> b.yearsOfStudent >> b.studentNumber;


   return ins;

}
```

4.3

```
#include <iostream>

#include <fstream>

#include <cstdlib>

#include "Bursary.h"


using namespace std;
```

```
int main()
{
   int year, modules;
   int studNo = 0;

   cout << "Enter student year :";
   cin >> year;
   cout << endl;

   cout << "Enter modules passed :";
   cin >> modules;
   cout << endl;

   Bursary criteria(studNo,year,modules);

   ifstream infile;

    infile.open ("Bursary.dat");

    if (infile.fail())
    {
      cout<<"Error opening file";
      exit(1); // for opening file"
    }

    Bursary candidate;

      while(infile >> candidate)
      {
          if (candidate == criteria)
              cout << candidate.get_Number() << endl;
```

```
        }
        infile.close();


    return 0;


}
```

4.4

A C++ friend functions are special functions which can access the private members of a class.

Friend functions have the following properties:

- 1) Friend of the class can be member of some other class.
- 2) Friend of one class can be friend of another class or all the classes in one program, such a friend is known as GLOBAL FRIEND.
- 3) Friend can access the private or protected members of the class in which they are declared to be friend, but they can use the members for a specific object.
- 4) Friends are non-members hence do not get “this” pointer.
- 5) Friends, can be friend of more than one class, hence they can be used for message passing between the classes.
- 6) Friend can be declared anywhere (in public, protected or private section) in the class.

Question 5

5.1

```
#ifndef TOPUP_H
#define TOPUP_H

#include "CellContract.h"

class Topup : public CellContract
{
    public:
        Topup();
        Topup(int minutesP, int dataP, smsP);
        void addAirtime(int talkTimeP);
        void addData(int dataP);

        void addSmsBundle(int smsP);

        void getBalances();


    private:

        int SMS;

};


#endif // TOPUP_H
```

5.2

```
Topup::Topup(int minutesP, int dataP, smsP) : CellContract(minutesP, dataP),SMS(smsP){}
```

5.3

Talktime and MBData are not member variable of Topup class.

5.4

No it is not because it has the same return type and parameter, rather it is overriding .

Question 6

6.1

```
template <TCashier, TPWord, TTerm>

class CashierList

{

    public:

        CashierList();

        void addOne(TCashier pcashier, const TPWord &pword, TTerm pterm);

        TPWord lookup(TCashier pcashier) const;


    private:

        vector <TCashier> cashier;

        vector <TPWord> password;

        vector <TTerm> terminal;

};
```

6.2

```
template <class TCashier, class TPWord, class TTerm>

void CashierList<TCashier, TPWord, TTerm>::addOne(TCashier pcashier, const TPWord &pword,
TTerm pterm){


  cashier.pushback(pcashier);

  password.pushback(pword);

  terminal.pushback(pterm);

}
```

6.3

```
CashierList<string,double,int> cashier1("A001", 55.22, 3);
```