1. typeA func(typeB a)
   {
     typeC b = 5;
     return b;
   }

   (a) (1 point) Will this function compile?  no
   (b) (3 points) Why or why not?  return type doesn’t match type of variable
    being returned

or

   (a) (1 point) Will this function compile?  yes
   (b) (3 points) Why or why not?  b will get cast to typeA (since they’re all
    integer types)

2. if (a<10 && b<5)
   {
     cout<<”hello”<<endl;
   }else if{a>10 || b >4}
   {
     switch(c)
     {
       case 1:
         cout<<”two”<<endl;
         break;
       case 2:
         cout<<”one”<<endl;
         break;
       default:
         cout<<”four”<<endl;
     }
   }else if((a-b)> 0 && d < 15)
   {
     if (d > 3)
       cout<<”five”<<endl;
     else
       cout<<”zero”<<endl;
   }
else
   cout<<”goodbye”;

What is the output of the above, given:

(a) (2 points) int a = 10; int b = 5; int c=1; int d=3;  two
(b) (2 points) int a = 5; int b = 2; int c=1; int d=5; hello
(c) (2 points) int a = 10; int b = 2; int c=2; int d=5; five
3. Machine epsilon is the smallest positive floating point number such that 1.0 + machine epsilon is not equal to 1.0. Epsilon can be approximated by repeatedly dividing a number by a multiple of 2.

(a) (10 points) Write a program to calculate machine epsilon.

```c
float machEps = 1.0;
do{
    machEps /= 2.0;
}while((1.0 + machEps) != 1.0);
```

(b) (10 points) Explain what kind of loop you used and why. **Do while (or while) because the loop should run until the condition is met. (Do while is preferred because the code should run at least once).**

4. A lookup table is an array (or other data structure) that stores precomputed values so that a value can be retrieved from memory instead of computed on demand. For example \{1,1,2,6,24,120\} would be the lookup table for factorials where the max lookup value is n=5.

(a) (10 points) Write a function that takes in an input n and returns a lookup table for factorials up to n such that facttable is an array where facttable[i] = i!

```c
void factorialTable(int facttable[], int n){
    facttable[0] = 1;
    for(int i=1; i<=n; i++)
        facttable[i] = (facttable[i-1] * i);
}
```

(b) (10 points) Explain how you would return the answer and why you would do it that way.

```c
int n = 15;
int facttable[n+1];
factorialTable(facttable, n);
//the following two lines are optional
//example of how you would use the table
i = 5;
cout << "lookup_val=" << i << " \"factorial\" = " << facttable[i] << endl;
```

**The array needs to be created before the function call and passed in by reference because there is no (clean) way to return an array.**
5. Write a coin flip simulation program. **Note: all 4 variants have almost the exact same code, just swap in T, B, or R respectively**

```c
#include <cstdlib>

char flipcoin()
{
    if (rand()/double(RAND_MAX) > .5)
        return 'H';
    return 'T';
}
```

(a) Write a function that simulates n coin flips by filling the array char coinflips[n] with random 'H's and 'T's. Pass in n as an argument and use the provided function flipcoin to obtain a random 'H' or 'T'.

```c
void getflips(char [], int);
void getflips(char coinflips[], int n){
    for (int i = 0; i < n; i++)
        coinflips[i] = flipcoin();
}
```

(b) Write a function that takes in an array of coinflips and calculates the probability of picking a H from the array of trials.

```
P(H) = \frac{\text{number of } H's}{\text{number of } \text{coinflips}}
```

```c
double probH(char [], int);
double probH(char coinflips[], int N){
    int Hcount = 0;
    for (int i = 0; i < N; i++)
        if (coinflips[i] == 'H')
            Hcount++;
    return (double) Hcount/N;
}
```

(c) (10 points) Implement a program that calculates and prints out the probability of getting heads in n trials. The user should input n, and the program needs to validate the input and only start the calculation once valid input is entered. The program should repeat until the user types in Q.

```c
int n;
char stop;

do{
    cout << "# of coin flips? ";
    ...
cin>>n;
while(n<1){
cout<<"Invalid input : (n<1)";
cin>>n;
}
char coinflips[n];
getflips(coinflips, n);
cout<<"P(H) : "<< probH(coinflips, n)<<endl;
cout<<"Hit Q to quit"<<endl;
cin>>stop;
}while(stop != 'Q' && stop != 'q');