Problem 1. To compute the probabilities using prior sampling, I used the following function.

```
def prior_sampling(samples):
  samples = cleanup_samples(samples)

  pc_t = [x for x in samples if x[0]]
  p1 = float(len(pc_t)) / 25.0

  pr_t = [x for x in samples if x[2]]
  pc_tgr = [x for x in pr_t if x[0]]
  p2 = float(len(pc_tgr)) / float(len(pr_t))

  pw_t = [x for x in samples if x[3]]
  ps_tgw = [x for x in pw_t if x[1]]
  p3 = float(len(ps_tgw)) / float(len(pw_t))

  pwc = [x for x in samples if (x[3] and x[0])]
  psgwc = [x for x in pwc if x[1]]
  p4 = float(len(psgwc)) / float(len(pwc))

  print("\nCOMPUTING PROBABILITIES VIA PRIOR SAMPLING:\n")
  print("P(C) = %.4f" % p1)
  print("P(C|R) = %.4f" % p2)
  print("P(S|W) = %.4f" % p3)
  print("P(S|C,W) = %.4f" % p4)

  return [p1,p2,p3,p4]
```

cleanup_samples is a function which takes the 100 random numbers and returns 25 samples. Each sample consists of four boolean values: C, S, R, and W, respectively. Here are the results:

```
P(C) = 0.4800
P(C|R) = 0.7500
P(S|W) = 0.4000
P(S|C,W) = 0.0000
```

Problem 2. Here are the exact values of the probabilities:

```
P(C) = 0.5000
P(C|R) = 0.8000
P(S|W) = 0.4737
P(S|C,W) = 0.0474
```

Here are the errors produced by prior sampling:

```
PRIOR SAMPLING ERROR FOR P(C): 0.0200
PRIOR SAMPLING ERROR for P(C|R): 0.0500
PRIOR SAMPLING ERROR FOR P(S|W): 0.0737
PRIOR SAMPLING ERROR FOR P(S|C,W): 0.0474
AVERAGE ERROR FOR PRIOR SAMPLING: 0.0478
```

Problem 3. To compute the probabilities using rejection sampling, I used the following function.

```
def rejection_sampling(samples):
  pc_t = [x for x in samples if x < .5]
  p1 = float(len(pc_t))/100.0

  pr_t = []
  for i in range(100):
    if i % 2 == 1:
      pr_t.append([samples[i-1],samples[i]])
  pr_t = [x for x in pr_t if (x[0] < .5 and x[1] < .8) or (x[0] >= .5 and x[1] < .2)]
  p2 = float(len([x for x in pr_t if x[0] < .5])) / float(len(pr_t))

  pr_w = [x for x in cleanup_samples(samples) if x[3]]
  p3 = float(len([x for x in pr_w if x[1]])) / float(len(pr_w))

  pr_cw = [x for x in cleanup_samples(samples) if x[0] and x[3]]
  p4 = float(len([x for x in pr_cw if x[1]])) / float(len(pr_cw))

  print("\nCOMPUTING PROBABILITIES VIA REJECTION SAMPLING:\n")
  print("P(C) = %.4f" % p1)
  print("P(C|R) = %.4f" % p2)
  print("P(S|W) = %.4f" % p3)
  print("P(S|C,W) = %.4f" % p4)

  return [p1,p2,p3,p4]
```

Here are the results:

```
P(C) = 0.4900
P(C|R) = 0.7037
P(S|W) = 0.4000
P(S|C,W) = 0.0000
```

Problem 4. The average error for rejection sampling was greater than the average error for prior sampling:

```
REJECTION SAMPLING ERROR FOR P(C): 0.0100
REJECTION SAMPLING ERROR FOR P(C|R): 0.0963
REJECTION SAMPLING ERROR FOR P(S|W): 0.0737
REJECTION SAMPLING ERROR FOR P(S|C,W): 0.0474
AVERAGE ERROR FOR REJECTION SAMPLING: 0.0568
```

Notice that the error for $P(S|W)$ and $P(S|C,W)$ are the same. This is because we only have four variables, and all four are relevant to compute the samples where W is positive. Therefore, even in rejection sampling we cannot exclude any of the four variables, which makes it identical to prior sampling. However, in the first two cases, we have something different.

For $P(C)$, we simply treat all 100 random numbers as a sample for C, since C is independent. This ended up giving us a more accurate approximation of $P(C)$.

For $P(C|R)$, we can treat each pair of numbers as a sample for $[C,R]$. This is because the probabilities of S and W don't affect the probabilities for R or for C. This time, we get 50 samples. However, the result still ends up being less accurate. We underestimated $P(C|R)$, which likely means our random number generator simply didn't give us enough samples where R was positive and C was positive.