```cpp
#include "Gate.h"

// Function prototype definitions to update each gate type
char andUpdate(char one, char two);
char orUpdate(char one, char two);
char nandUpdate(char one, char two);
char norUpdate(char one, char two);
char xorUpdate(char one, char two);
char xnorUpdate(char one, char two);
char notUpdate(char one, char two);

// Gate constructor
Gate::Gate(Wire* in1, Wire* in2, Wire* out, int typ, int dly)
{
        input1 = in1;
        input2 = in2;
        output = out;
        type = typ;
        delay = dly;
}

// Gate destructor
Gate::~Gate()
{
    delete input1;
    delete input2;
    delete output;
}


// Gate update function
void Gate::update(int time, priority_queue<Event>* eq, int priority)
{
    // Bringing in the current wire values
    char one = (*input1).getValue();
    char two = (*input2).getValue();
    char ret;
    // Checking the gate type and calling the appropriate update function
    switch(type)
    {
        case 1:
        ret = andUpdate(one, two);
        break;
        case 2:
        ret = orUpdate(one, two);
        break;
        case 3:
        ret = nandUpdate(one, two);
        break;
        case 4:
        ret = norUpdate(one, two);
        break;
        case 5:
        ret = xorUpdate(one, two);
        break;
        case 6:
        ret = xnorUpdate(one, two);
        break;
        case 7:
        ret = notUpdate(one, two);
        default:
        break;
    }

    // Creating a new event to change the output wire
    // after the appropriate gate delay
```

```
67      Event newEvent(time+delay, output, ret, priority);
68      (*eq).push(newEvent);
69  }
70
71  // Update functions for each type of gate
72  char andUpdate(char one, char two)
73  {
74      if((one == 'x' || two == 'x')&&(one == '-' || two == '-'))
75      {
76          return 'x';
77      }
78      if(one == '-' && two == '-')
79      {
80          return '-';
81      }
82      return '_';
83  }
84
85  char orUpdate(char one, char two)
86  {
87      if((one == 'x' || two == 'x')&&(one == '_' || two == '_'))
88      {
89          return 'x';
90      }
91      if(one == '-' || two == '-')
92      {
93          return '-';
94      }
95      return '_';
96  }
97
98  char nandUpdate(char one, char two)
99  {
100     if((one == 'x' || two == 'x')&&(one == '-' || two == '-'))
101     {
102         return 'x';
103     }
104     if(one == '-' && two == '-')
105     {
106         return '_';
107     }
108     return '-';
109 }
110
111 char norUpdate(char one, char two)
112 {
113     if((one == 'x' || two == 'x')&&(one == '_' || two == '_'))
114     {
115         return 'x';
116     }
117     if(one == '_' && two == '_')
118     {
119         return '-';
120     }
121     return '_';
122 }
123
124 char xorUpdate(char one, char two)
125 {
126     if(one == 'x' || two == 'x')
127     {
128         return 'x';
129     }
130     if(one == '-' && two == '_')
131     {
132         return '-';
```

```
    }
    if(one == '_' && two == '-')
    {
        return '-';
    }
    return '_';
}

char xnorUpdate(char one, char two)
{
    if(one == 'x' || two == 'x')
    {
        return 'x';
    }
    if(one == '-' && two == '_')
    {
        return '_';
    }
    if(one == '_' && two == '-')
    {
        return '_';
    }
    return '-';
}

char notUpdate(char one, char two)
{
    if(one == 'x')
    {
        return 'x';
    }
    if(one == '-')
    {
        return '_';
    }
    if(one == '_')
    {
        return '-';
    }
}
```