1. (5 points) What is a template in C++? Describe an example when a C++ template would be useful?

A template in C++ is a mechanism for specifying the type (or types) handled by a class (or function) when the class (or function) is declared (or called), the type is specified. This allows a single implementation of an object to be used to store a wide range of types. For example, a stack template could be used to hold integers, doubles, Node objects, etc. It would be worthwhile to create a template when the same abstract data type (ATD) is needed for two or more different types. For example; when you need a stack of doubles AND a stack of integers.

2. (5 points) Describe the functions cin.peek(), cin.ignore(), and isspace().

- cin.peek() returns the next character in the standard input stream. It does not read the character (that is, it does not advance the file pointer).
- cin.ignore() skips the next character in the standard input stream. It is similar to reading the next character but the character is ignored.
- isspace() takes a single character argument. It returns true if that character is a space, tab, newline, or carriage return. It returns false otherwise.

3. (5 points) Describe how to use GDB to find the location of a Segmentation Fault.

1) Compile the program using the g option to g++ (use g for all files)
2) start the debugger and pass the program's executable as a command line argument:
   $ gdb a.out
3) type run at the gdb prompt:
   (gdb) run
4) when the segmentation fault occurs type here at the gdb prompt
   (gdb) where

4. (10 points) What are the similarities and differences between a stack and a queue? Give an example of a problem solved naturally using a stack as well as one using a queue.

A stack is a data structure in which the first object inserted into the stack (often called pushed onto the stack) is the last object removed from the stack (often called popped off of the stack) FILO.

Problems naturally solved using a stack: postfix calculator, runtime stack, breadth-first traversal of graphs. Stacks are great for REVERSING DATA

A queue is a data structure in which the first object inserted into the queue (often called enqueue) is the first item removed from the queue (often called dequeue) FIFO.

Problems naturally solved using a queue: event based simulation, resource allocation (e.g. printer queue, runtime queue (the queue for the CPU))
5. (10 points) What does the following program print. Show the steps of the calculation for ANY credit.
   Be very methodical, it is easy to make a mistake.
   ```cpp
#include <iostream>
using namespace std;
int f(int n[], int size)
{
    if (size <= 1)
        return 1;
    return (n[0] + n[size-1]) * f(n+1, size-2);
}

int main()
{
    int nums[] = {1,2,3,4}; // initializes array so first element is 1, 2nd is 2, etc
    cout << f(nums, 4) << endl;
}
```
   It prints 25
   f() is called in this order:
   f([1,2,3,4], 4) = (1 + 4) * f([2,3], 2)
   f([2,3], 2) = (2 + 3) * f([], 0)
   f([], 0) = 1
   replacing the values for function calls in the above lines we get
   f([2,3], 2) = (2 + 3) * 1 = 5 * 1 = 5
   f([1,2,3,4], 4) = (1 + 4) * 5 = 5 * 5 = 25

6. (10 points) Write the recursive function product(). This function recursively calculates the product of all the elements in an array. Assume the given array has at least 1 element. Think carefully about the base case.
   ```cpp
int product(int array[], int size)
{
    assert(size >= 1);
    if (size == 1)
        return array[0];
    return array[0] * product(array+1, size-1);
}
```
7. (10 points) Using the class Pqueue definition on the last page, write ~Pqueue().
Pqueue::~Pqueue()
{
    while ( dequeue() )
    {
    }
}

8. (10 points) Using the class Pqueue definition on the last page, write dequeue(). Return
a pointer to the Cust object at the front of the queue. If the queue is empty, return NULL.
Make sure your solution works with the enqueue() you write in the next question.
Cust *Pqueue::dequeue()
{
    if (m_head == NULL)
        return NULL;
    Cust *cust = m_head->m_cust;
    Node *tmp = m_head;
    m_head = m_head->m_next;
    delete tmp
    return cust;
}

9. (15 points) Using the class Pqueue definition on the last page, write the function
enqueue(Cust *cust, int priority) which enqueues the given Cust object on to the queue
using the given priority. The new element should be inserted so that this function works
with dequeue() in the previous question.
void Pqueue::enqueue(int priority, Cust *cust)
{
    // if new item belongs at front of queue
    if (m_head == NULL || priority < m_head->m_priority)
    { m_head = new Node(priority, cust, m_head); }
    else
    {
        // find the node to insert after
        Node *ptr = m_head;
        while (ptr->m_next && priority >= ptr->m_next->m_priority)
        { ptr = ptr->m_next; }
        assert(ptr != NULL);
        ptr->m_next = new Node(priority, cust, ptr->m_next);
    }
}