This exam is closed book and notes. There are 4 parts for a total of 100 points. Answer them all on the question sheets. Incomprehensible answers get zero points!

**Part 1 (20 points)**

Complete the C++ function declared `char *reverse(const char *pch);` so:

1. (4 points) It will call function `size_t strlen(const char *)` to determine the length of the C-string whose address is the value of parameter `pch`, and save this length in variable `mylength`. (Remember that a “C-string” is a null-terminated array of `char`.)

2. (4 points) It will allocate a dynamic array of `char` just big enough to hold a C-string with the length determined above, and store its address in the variable named `pnewch`.

3. (10 points) It will store into this dynamic array the C-string obtained by **reversing the order of the characters of the given C-string** (from `pch[]`). (Example: Reversing "ABC123Z" gives "Z321CBA".)

4. (2 points) The function should return the address of the reversed string.

```c++
#include <cstring> // Supplies strlen()
#include <cstdlib> // Supplies size_t
using namespace std;
char * reverse( const char *pch )
{
    size_t mylength;
    char * pnewch;
    // Your code here
}
```
**Part 2 (20 points)**

The following defines a class that implements a dynamically allocated partially filled array of floats data structure:

```cpp
#include <stdlib> // Supplies size_t
using namespace std;

class Part2Array
{
  public:
    Part2Array() //Constructor.
    { data = new float[13];
      capacity = 13;
      used = 0;
    }
    void remove(size_t i);
    //Precondition: 0<=i<=used-1
    //Postcondition: The entry in position i has been removed and all used
    // entries right of position i are shifted left so the used entries
    // continue to be stored in a prefix of the array pointed to by data.
    void reserve(size_t new_capacity);
    //Precondition: new_capacity >= used
    //Postcondition: The capacity of the dynamically allocated array is
    // now new_capacity; and the original contents are preserved.
    ....
    //Various data access functions are not shown.
  private:
    float * data; //Invariant: data!=NULL.
    size_t used; //Invariant: The array prefix used is data[0..(used-1)].
    size_t capacity; //Invariant: The number of allocated elements=capacity.
};
```

Write complete and correct implementations of the member functions (a) `remove` and (b) `reserve`. 

Part 3 (25 points)

The node type for a singly-linked-list of floats is defined by:

```c
struct node
{
    float data;
    node *fore; //fore==the address of the next node if any, NULL if not.
};
```

Suppose we have one singly-linked-list whose head and tail node addresses are respectively stored in variables HEAD and TAIL. Suppose cursor points to some node, not the last.

```c
node * HEAD;
node * TAIL;
node * cursor;
```

Assume all nodes are dynamically allocated and should be returned to the “heap” or free-store when they are deleted. Complete the code below to delete (from this singly-linked list) the node **just after the node** pointed to by cursor:

But first answer here (for 5 points, in English, not C++): What has to be done in the special case when the node to be deleted is the last node in the list?

```c
assert( cursor != NULL && cursor != TAIL );
(write the code below..)
```
Part 4 (35 points, 7 questions)

1. Draw a “memory picture” or data structure diagram (with boxes, variable names, arrows and values) for what is in memory after the code sequence below finishes running:

   ```c
   int I = 5;
   int J = 63;
   int K;
   int * P;
   int * Q;
   P = &I;
   Q = &J;
   K = *Q;
   *Q = *P;
   P = Q;
   Q = new int;
   *Q = 98;
   ```

   If you made changes as you worked out this problem (which indeed you SHOULD do), **mark clearly** what your **final picture** is! (Ambiguities get 0 points).

2. Why don’t linked lists make use of an operation like “reserve” described in Part 2?

3. Bozo Simpleton, who didn’t understand how to apply “.h” files, wrote a class definition for class BoSim near the top of the same file in which he wrote the implementations of class BoSim’s member functions. He named that file “BoSim.cxx”.

   He eventually corrected some syntax errors and got this implementation file to compile.

   He then tried to compile the “main” module (in a file named main.cxx) that declared some variables of type BoSim. Since the compiler reported errors, he wrote the definition of class BoSim near the top of main.cxx. Finally main.cxx was compiled, the linking was successful, and the program seemed to run OK.

   Explain why what Bozo did is poor software development practice, in terms of what might go wrong in the future.
4. This question is about the Project 2 : in file <filename> command. Suppose the file was opened successfully by using
\texttt{ifstream \texttt{myfile(filename)};} Each line from the file is read using \texttt{myfile.getline(...) as the condition-expression in a while( )\{}}
or \texttt@if( )\{\}} statement. The return value of \texttt{myfile.getline(...) will evaluate to false in two situations. Explain each situation:}
\begin{itemize}
\item[\textbf{Situation 1:}] 
\item[\textbf{Situation 2:}] 
\end{itemize}

5. In this course’s project and laboratory assignments, what is \texttt{build.sh} for?

6. Explain one advantage of using a C-string, \texttt{instead of} a partially filled \texttt{char} array together with a \texttt{size_t “used” variable.}

7. Choose \texttt{one} operation that takes constant time on a linked list but not on a partially filled array (1 pt.). Explain why the amount of time used for this operation is constant with the linked list but is not constant with the partially filled array (4 pt.).