CS140 Homework 1 [10 pts]

Due 11.10am Tuesday January 22, 2019

Problem 1a [3]

Write a C++ program called hw_1a.cpp that prints the value of argc followed by the addresses of each argv argument as well the corresponding (C-style) strings. The program must contain everything needed to compile. The following is an example output:

```
unix> ./hw_1a This is great
Num args = 4
argv[0] = 0x7fffeb3fb1f8 ./hw_1a
argv[1] = 0x7fffeb3fb200 This
argv[2] = 0x7fffeb3fb208 is
argv[3] = 0x7fffeb3fb210 great
```

Problem 1b [7]

Copy hw_1a.cpp into a new program called hw_1b.cpp that computes and prints the length of each command line argument. See an output example below. Do not use functions from <string> or <cstring>. Instead write your own function "int strlen(char *)" which is given a pointer to a C-style string, namely, argv[i], and returns the length thereof (number of characters). Use a pointer to advance through the string. Use pointer dereferencing to determine when to stop. Hint: C-style strings are NULL-terminated meaning the last character equals '\0'; the condition *s=='\0' is thus met when the end of the string has been reached. Hint: See the pointer_handout for a related function that compares two C-style strings.

```
unix> ./hw_1b I learn so much!
Num args = 5
argv[0] = 0x7fff8b27e098 ./hw_b (strlen=6)
argv[1] = 0x7fff8b27e0a0 I (strlen=1)
argv[2] = 0x7fff8b27e0a8 learn (strlen=5)
argv[3] = 0x7fff8b27e0b0 so (strlen=2)
argv[4] = 0x7fff8b27e0b8 much! (strlen=5)
```

Submission

Submit your answer via Canvas. If you only complete Problem 1a, submit hw_1a.cpp. If you also complete Problem 1b, submit only hw_1b.cpp as it supersedes hw_1a.cpp.

Submission will close automatically when the deadline rolls around to allow discussion of the answer in class. If you worry that you might miss the deadline, submit your answer early. This will be standard procedure going forward and will not be mentioned again.

CS140 Homework 1 [10 pts]

Problem 1a

Problem 1b

```
#include <iostream>
using namespace std;

int strlen(char *s) {
   int len=0;
   while (*s++)
      len++;
   return len;
}

int main(int argc, char *argv[]) {
   cout << "Num args = "
          << argc
          << "\n";
   for (int i=0; i<argc; i++)
      cout << "argv["
             << i  << "] = "
             << &argv[i]
             << " "
             << argv[i]
             << " (strlen="
             << strlen(argv[i])
             << ")\n";
 }
```

CS140 Homework 2 [10 pts]

Due 11.10am Thursday January 24, 2019
Late submissions will not be accepted

Problem 1

Write single linked list class member functions push_front, pop_front and front which work just like the back equivalents on the list code handout, only the front of the list is used instead of the back. The function signatures and the basic modes of operation can be described as follows:

```
void push_front(const int &) { insert(0, din); }
void pop_front() { erase(0); }
const int &front() { node *p=findnode(0); return p->data; }
```

Your job is to implement each of these functions without calling other list member functions. Instead make explicit use of pointers when relinking and accessing the nodes. Submit only the code for these three functions.

Hint: Copy the code from the above mentioned list member functions. Remove any code not needed for the special case of operating at the front of the list, i.e., when inserting, removing and accessing the first node reachable from the head node of the list. Then clean-up and test the resulting code.

Hint: Sketch what you are trying to do on a piece of paper. Don't try to "model" it all in your head.

CS140 Homework 2 [10 pts]

Problem 1

```
void list::push_front(const int & din) {
   node *p = new node(din);
   p->next = head->next;
   head->next = p;
   N++;
}

void list::pop_front() {
   node *p = head->next;
   if (p) {
      head->next = p->next;
      delete p;
      N--;
   }
}

const int & list::front() {
   node *p = head->next;
   if (p) return p->data;
   else   return head->data;
}
```

CS140 Homework 3 [10 pts]

Due 11.10am Tuesday January 29, 2019

Problem 1

See list_handout.pdf for the class definition and implementation of a double linked list. Rewrite the findnode(int index) function to perform a forward search in the first half of the list if index<N/2 and a backward search in the second half of the list otherwise. The result should be that no more than N/2 nodes will be visited as part of a search.

Hint: Sketch what you are trying to accomplish on a piece of paper before you think about the code.

CS140 Homework 3 [10 pts]

Problem 1

```
node *list::findnode(int i) {
   if (i == -1)    return head;
   if (i == N-1) return head->prev;

   node *p;

   if (i < N/2) {
      p = head->next;
      while (i--)
         p = p->next;
     } else {
      p = head->prev;
      i = N-1-i;
      while (i--)
         p = p->prev;
   }

   return p;
}
```

CS140 Homework 4 [10 pts]

Due 11.10am Thursday January 31, 2019

Problem 1 [7]

See the stack_handout for the class definition and wrapper implementation of a stack based on a list. Your task is to replace the wrapper implementation with code based on a dynamically allocated array. Assume the data stored on the stack are simple integers (int). Implement the public member functions stack(), ~stack(), empty(), size(), push(), pop() and top(). Single line functions may be defined within the class if you like. Multi-line functions should be defined outside of the class definition. The constructor initializes all member variables and allocates memory for the array. The destructor deallocates the array memory. Make the size of the array 10. Do not include resizing. Do not include any error checking.

Problem 2 [3]

What type of error checking should member functions push(), pop() and top() include if the code were to be used in real application?

CS140 Homework 4 [10 pts]

Problem 1

```
class stack {
   public:
      stack();
      ~stack() { if (data) delete [ ] data; }

      bool empty() const { return N == 0; }
      int  size()  const { return N; }

      void push(const int &din) { data[N++] = din; }
      void pop() { N--; }   OR  pop() { data[--N] = 0; }
      const int & top() { return data[N-1]; }

   private:
      int N;
      int Nmax;
      int *data;
};

stack::stack() {
   N = 0;
   Nmax = 10;
   data = new int[Nmax];
   for (int i=0; i<Nmax; i++)
      data[i] = 0;
}
```

Problem 2

push() error checking: if (N==Nmax), capacity of array has been reached.

pop() error checking: if (N==0), there is no element to pop.

top() error checking: if (N==0), there is no data to return.

CS140 Homework 5[10 pts]

Due 11.10am Tuesday February 5, 2019

Problem 1

Rewrite the stats class from Prog1c in Lab 1 to be template based. To make things a little easier, delete the min-max computation. N remains an integer but all data references are to be made template based. Test your program for different data types, e.g., int, float and string.

Hint: Replace the sum=0 initialization with sum=T() to automatically obtain the default value of 0 for T being an int, 0.0 for a float and "" for a string.

Submit a single .cpp program that includes the class definition, the source code implementing it, and a simple driver code that tests it.

CS140 Homework 5 [10 pts]

Problem 1

```
template <typename T>
class stats {
   public:
      stats();
      void push(T &);
      void print();

   private:
      int N;
      T sum;
};

template <typename T>
stats<T>::stats() {
   N = 0;
   sum = T();
}

template <typename T>
void stats<T>::push(T &x) {
   N++;
   sum += x;
}

template <typename T>
void stats<T>::print() {
   cout << "N   = " << N << "\n"
        << "sum = " << sum << "\n";
}

int main( ) {
   stats<int> S;
   int x;

   while (cin >> x)
      S.push(x);

   S.print();
}
```

CS140 Homework 6 [10 pts]

Due 11.10am Thursday February 7, 2019

Problem 1 [5]

See the code below. Write a template based implementation of function ex_count which counts the number of instances of the first element in a sequence. For example, if the input is "1 3 2 4 3 1 4 1", the output is "Found 3 instances of target 1". Hint: Seek inspiration from the STL_intro_handout.

```
#include <iostream>
#include <list>
using namespace std;

int main() {
  list<int> v;
  list<int>::iterator iv;
  int value;
  while (cin >> value)
    v.push_back(value);

  int target = *v.begin();
  int N = ex_count(v.begin(), v.end(), target);

  cout << "Found " << N << " instances of " << target << "\n";
}
```

Problem 2 [5]

Rewrite template based function ex_count( ) from Problem 1 to take an ex_eq function object as its third argument instead of the integer target. See the changed line of code below that must now be supported. Write a template based class implementation of function object ex_eq that compares data elements against the target provided when instantiated and returns true if the two are equal. The output is the same as above. Hint: Seek inspiration from the STL_intro_handout.

```
int N = ex_count(v.begin(), v.end(), ex_eq<int>(target));
```

Extra Challenge [0]

Declare the ex_eq target to be a const and work out how to initialize it. Hint: Look up the so-called colon operator as the value of a const cannot be changed using the assignment operator.

CS140 Homework 6 [10 pts]

Problem 1

```
template <typename iT, typename T>
int ex_count(iT i1, iT i2, const T & target) {
  int N=0;

  while (i1 != i2) {
    if (*i1 == target)
      N++;
    ++i1;
  }

  return N;
}
```

Problem 2

```
template <typename T>
class ex_eq {
  public:
    ex_eq(const T &new_target) : target(new_target) {}
    bool operator() (const T &n) const { return n == target; }

  private:
    const T target;
};

template <typename iT, typename Function>
int ex_count(iT i1, iT i2, Function istrue) {
  int N=0;

  while (i1 != i2) {
    if (istrue(*i1))
      N++;
    ++i1;
  }

  return N;
}
```