CMPT 125 D100, Fall 2025

Final Exam - SOLUTIONS
December 15, 2025

Name_____

SFU ID: |__|__|__|__|__|__|__|__|__|

Problem 1	
Problem 2	
Problem 3	
Problem 4	
TOTAL	

Instructions:

1. The duration of the exam is 180 minutes.
2. Write your full name and SFU ID **clearly**.
3. This is a closed book exam, no calculators, cell phones, or any other material.
4. The exam consists of four (4) problems.
5. Write your answers in the provided space.
6. There is an extra page at the end of the exam. You may use it if needed.
7. **Explain all your answers.**
8. **Really, explain all your answers.**

Good luck!

Problem 1 [25 points]

a) [2 points each] Suppose you have a program in C named *program.c*.

1. What is the command in Linux to compile it into an executable file?

   Answer: gcc program.c

2. What is the default name of the compiled executable?

   Answer: a.out

3. How can you specify the name of the executable in the compilation line?

   Answer: gcc program.c -o specific_name

b) [5 points] Will the code below compile?
If yes, what will be the result of the execution? If not, explain errors/warnings/potential issues.

```
#include <stdio.h>

int array_sum(const int arr[], int len) {
  return arr[len]+arr[len+1]+arr[len+2];
}

int main() {
  const int arr[] = {5,4,3,2,1};
  array_sum(arr, 2);
  return 0;
}
```

Answer: yes, it will compile. array_sum will compute arr[2]+ arr[3]+arr[4]=3+2+1=6
The program does not print anything.

c) [7 points] What is the running time of foo(n)? Use big-O notation to express your answer. Explain your answer.

```
int foo(int n) {
  int sum=0;
  for(int i=0; i<n; i++)
    for(int j=1; j<i; j=j*2)
      sum+=j;
  return sum;
}
```

Answer:
The outer loop has n iterations.
In each iteration of the outer loop, the inner loop makes $\log_2(i) < \log_2(n)$ iterations, each iteration running in O(1) time.
Therefore, the total running time is $n^* \log_2(n)^*O(1) = O(n \log(n))$

d) [7 points] What is the running time of foo(n)? Use big-O notation to express your answer. Explain your answer.

```
int foo(int n) {
  int sum=0;
  for(int i=1; i<n; i=i*2)
    for(int j=0; j<i; j++)
      sum++;
  return sum;
}
```

Answer:
The outer loop has log(n) iterations: $i=1,2,4,8,16\ldots2^k$ for k such that up $2^k<n$.
In each iteration of the outer loop, the inner loop runs for i steps.
Therefore, the total running time is $1+2+4+8+16+\ldots + 2^k$ + up to n.
This is a geometric series, and the sum is at most $2^{k+1}-1 < 2n = O(n)$

Problem 2 [25 points]

a) [13 points] Write a function that gets a string, and reverses each word in the string. Words in the string are substrings separated by spaces ' '. Note that there might be several consecutive spaces in a string.
For example,

- If we apply it on s="ABCD XYZ",
  then s becomes "DCBA ZYX".
- If we apply it on s=" 12 A bCde ",
  then s becomes " 21 A edCb ".
- If we apply it on s="do you like rigatoni pasta?",
  then s becomes "od uoy ekil inotagir ?atsap".

```
void reverse_each_word(char* str) {
```

Answer: We will need a helper function that reverses the substring srt[start…end]

```
    void reverse_word(char* str, int start, int end) {
      char tmp;
      while (start<end) {
        tmp = str[start];
        str[start] = str[end];
        str[end] = tmp;
        start++;
        end--;
      }
    }
```

We solve the question by iterating through the array and reversing each word using the helper function.

```
    int i = 0;
    while (str[i]) { // iterate until end of string
      while (str[i]==' ')  // skip all spaces
        i++;
      if (str[i]) { // if not end of string, reverse the next word
        int start = i;
        while (str[i]!='\0' && str[i]!=' ' )
          i++;
        reverse_word(str, start, i-1)
      }
    }
}
```

b) [12 points] Write a function that gets a string with an arithmetic expression that contains a positive number, followed by an operator ('+' '-' '*' '/'), followed by a positive number, and returns the answer to the computation as an int. For example:

- on input `"123+59"` the output should be 182.
- on input `"53*2"` the output should be 106.
- on input `"600-9"` the output should be 591.
- on input `"432/4"` the output should be 108.

You may assume the input is in the correct format: numbers are positive, no spaces, etc.
You may assume that all numbers fit into an int.
You may use the standard functions if necessary (e.g., use the library string.h).

```
int simple_calc(const char* str) {


  int i = 0;
  int x = 0; // compute the first term
  while (str[i] >= '0' && str[i] <= '9') {
    x = x*10 + str[i] - '0';
    i++;
  }

  char op = str[i];  // save the operation
  i++;

  int y = 0;  // compute the second term
  while (str[i] >= '0' && str[i] <= '9') {
    y = y*10 + str[i] - '0';
    i++;
  }


  if (op == '+') return x+y;
  if (op == '-') return x-y;
  if (op == '*') return x*y;
  if (op == '/') return x/y;









}
```

Problem 3 [25 points]

a) [12 points] Consider the following definition of a Linked List of ints.

```
struct LL_node {
  int data;
  struct LL_node* next;
};
typedef struct LL_node LL_node_t;

typedef struct {
  LL_node_t* head;
} LL_t;
```

Write a function that gets two linked lists and checks if they have exactly the same values in the same order.

- If the lists are equal the function returns -1.
- Otherwise, it returns the first index (starting from 0), where the two lists are not equal.

For example,

- If list1 = 0->1->**7** and list2 is 0->1->**4**, the function returns 2.
- If list1 = 8->**4**->9 and list2 is 8->**5**->7, the function returns 1.
- If list1 = 0->1->2->3->**4**->10 and list2 is 0->1->2->3, the function returns 4.

```
int LL_equal(LL_t* list1, LL_t* list2) {
  LL_node_t* node1 = list1->head;
  LL_node_t* node2 = list2->head;
  if (node1 == node2)
    return -1;

  int index = 0;
  while (node1 != NULL && node2 != NULL) {
    // iterate through both lists
    if (node1->data != node2->data)
      return index;
    node1 = node1->next;
    node2 = node2->next;
    index++;
  }
  // reached the end of one of the lists
  if (node1 == NULL && node2 == NULL)
    return -1;   // lists are equal
  else
    return index; // one list is longer than the other
}
```

b) [13 points] Consider a stack of ints with the following operations.

```
typedef struct stack {
  ...
} stack_t;

stack_t* create_empty_stack();
void stack_push(stack_t* s, int item);
int stack_pop(stack_t* s);
bool is_empty(stack_t* s);
void stack_free(stack_t* s);
```

* Note that the exact implementation of stack_t is unknown.

Write a function that gets a pointer to stack, and returns a new copy of the stack. The copy contains exactly the same elements in the same order.

```
stack_t* stack_copy(stack_t* s) {


  stack_t* ret = create_empty_stack();
  stack_t* tmp = create_empty_stack();

  while (!is_empty(s))     // save s into tmp
    stack_push(tmp, stack_pop(s));
  // s is now empty

  // copy from tmp into s and ret at the same time
  while (!is_empty(tmp)) {
    int n = stack_pop(tmp);
    stack_push(s, n);
    stack_push(ret, n);
  }

  stack_free(tmp);
  return ret;









}
```

Problem 4 [25 points]

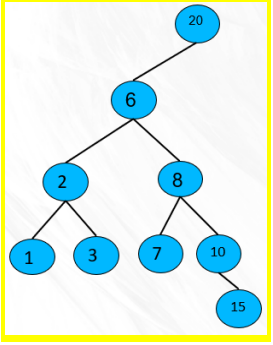
In this problem use the following struct for Binary Tree of ints.

```
struct BTnode {
  int value;
  struct BTnode* left;
  struct BTnode* right;
  struct BTnode* parent;
};
typedef struct BTnode BTnode_t;
```

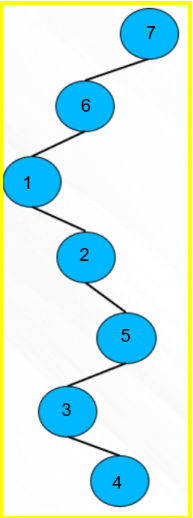
a) [5 points] Draw a Binary Search Tree whose *preorder* traversal is [20,6,2,1,3,8,7,10,15].

Answer: root is 20. All other numbers are less than 20, so it has only the left child.

- 6 is the left child. We have
  - [2,1,3] on the left of 6 -- construct the subtree with 2,1,3
  - [8,7,10,15] on the right of 6.
    - construct the tree with root=8, 7 on the left and [10,15] on the right

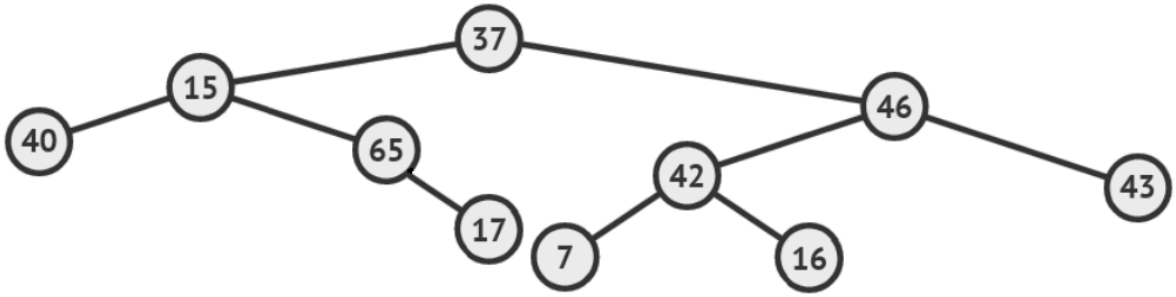


b) [5 points] Assign the numbers 1,2,3,4,5,6,7 to the vertices of the tree, so that the tree is a Binary Search Tree. Explain your answer



Answer:

c) [15 points] Write a function that gets a node in a binary tree, and returns the next node in the *inorder* traversal of the tree. For example, consider the following tree.



The inorder traversal of this tree is [40, 15, 65, 17, 37, 7, 42, 16, 46, 43]

- On input 40 the output needs to be the node containing 15.
- On input 15 the output needs to be the node containing 65.
- On input 65 the output needs to be the node containing 17.
- On input 43 the output needs to be NULL.

Explain your answer before writing the code.

```
BTnode_t* next_inorder(BTnode_t* node) {

  // if node has right child, go to the right subtree
  if (node->right) {
    BTnode_t* tmp = node->right;
    while (tmp->left)
      tmp = tmp->left;
    return tmp;
  }

  // node does not have the right child
  // go up as long as node is the right child of its parent
  while (node->parent && node->parent->right == node)
    node = node->parent;
  return node->parent;
```

Comment: this question was in assignment 4.

```
}
```

Extra page

Empty page