National Institute of Technology, Calicut

Department of Computer Science and Engineering

CS2092 Programming Lab

Assignment 3 - Extra Questions

---

1. Given two polynomials represented by two linked lists L and R respectively, write a program to add two polynomials. Each node in the list contains two data fields, ***coeff*** and ***degree*** (which are integers) and a link (pointer) to the next node. Pointers named 'head1' and 'head2' are used to point to the first nodes of L and R respectively. The pointer field of last nodes of L and R is set to NULL. Display the two polynomials and their sum. The input should be read from the file ***input.txt*** and output should be written to the file ***output.txt***. Your program must implement the following functions:

main() - repeatedly read a character 'l', 'r', 'p', 'a' or 's' from the file and call the given functions appropriately until character 's' is entered.

create(L, fp1) - read the *coeff* and *degree* of different terms of a polynomial from the input file pointed by file pointer ***fp1*** and create linked list ***L***.

print(L, fp2) - print the polynomial stored in a linked list ***L*** to the output file pointed by file pointer ***fp2***.

add_polynomial(L, R) - add two polynomials stored in linked lists ***L*** and ***R***.

Note: During addition of two polynomials, add the polynomial coefficients which have the same degree.

Input format:

a) The input consists of multiple lines. Each line contains a character from {**'l'**, **'r'**, **'p'**, **'a'**, **'s'**} followed by zero or more integers.
b) Character **'l'**: Character 'l' is followed by *coeff*, *degree* pairs of the terms (separated by a comma) of the first polynomial, where the pairs of each term are separated by a space.
c) Character **'r'**: Character 'r' is followed by *coeff*, *degree* pairs of the terms (separated by a comma) of the second polynomial, where the pairs of each term are separated by a space. (***Refer the sample input.***)
d) Character **'p'**: Character 'p' is followed by an integer from {1, 2, 3}.
    - If character 'p' is followed by 1, print the first polynomial.
    - If character 'p' is followed by 2, print the second polynomial.
    - If character 'p' is followed by 3, print the sum of first and second polynomial. ***Polynomials should be printed in the format given in sample output.***
e) Character **'a'**: Add the first and second polynomial.
f) Character **'s'**: Terminate the program.

Output format:

The output (if any) of each command should be printed on a separate line.

Sample Input:

```
l 5,2 3,1 2,0
r 6,3 7,2 3,0
p 1
p 2
a
p 3
s
```

Sample Output:

```
5x^2 + 3x + 2
6x^3 + 7x^2 + 3
6x^3 + 12x^2 + 3x + 5
```

2. Given two sorted singly linked lists A and B, write a program to merge the elements of A and B. Assume that the elements of A and B are in ascending order. The elements of the merged list should also be in ascending order. Each node in A and B has a *data* part (which is an integer) and a *pointer* to the next node. HEAD1 and HEAD2 are pointers that point to the start nodes of A and B respectively. The input should be read from the file ***input.txt*** and output should be printed to file ***output.txt.*** Your program must implement the following functions:

main() - repeatedly read a character 'a', 'b', 'p', 'm' or 's' from the file and call the given functions appropriately until character 's' is entered.

create(A, fp1) - read the elements of the linked list from the input file pointed by file pointer ***fp1*** and create linked list ***A***.

print(A, fp2) - print the elements of the linked list ***A*** to the output file pointed by file pointer ***fp2***.

merge(A, B) - merge the elements of linked lists ***A*** and ***B*** in sorted order.

Note: Do not use sort function

Input format:

a) The input consists of multiple lines. Each line contains a character from **{'a'**, **'b'**, **'p'**, **'m'**, **'s'}** followed by zero or more integers.
b) Character **'a'**: Character 'a' is followed by integers which are the elements of the linked list A, separated by a space. Create linked list A with these integers as elements.
c) Character **'b'**: Character 'b' is followed by integers which are the elements of the linked list B, separated by a space. Create linked list B with these integers as elements.
d) Character **'p'**: Character 'p' is followed by an integer from {1, 2, 3}.
    - If character 'p' is followed by 1, print the elements of linked list A, separated by a space.
    - If character 'p' is followed by 2, print the elements of linked list B, separated by a space.
    - If character 'p' is followed by 3, print the elements of the merged list, separated by a space.
e) Character **'m'**: Merge the elements of linked lists A and B in sorted order.
f) Character **'s'**: Terminate the program.

Output format:

The output (if any) of each command should be printed on a separate line.

Sample Input:

```
a 1 2 3

b 2 3 4

p 1

p 2

m

p 3

s
```

Sample Output:

```
1 2 3

2 3 4

1 2 2 3 3 4
```

3. You are given a queue Q of n elements. Write a program to reverse the first k elements in the queue Q, leaving the other elements in the same relative order. The input should be read from the file ***input.txt*** and output should be printed to file ***output.txt.*** (***Hint: Use a temporary stack.***)

Your program should include the following functions.

main() - repeatedly read a character 'c', 'p', 'm' or 's' from the file and call the given functions appropriately until character 's' is entered.

Enqueue(Q, element) - insert the data specified by ***element*** into queue ***Q***.

Reverse_k_elements(Q, k) - reverse the first ***k*** elements in the queue ***Q***, leaving the other elements in the same relative order.

print(Q, fp) - print the elements of the queue ***Q*** to the output file pointed by file pointer ***fp***.

Input format:

a) The input consists of multiple lines. Each line contains a character from **{'c'**, **'p'**, **'m'**, **'s'}** followed by zero or more integers.
b) Character **'c'**: Character 'c' is followed by an integer ***n***, which is the size of the queue. Next line contains ***n*** integers, separated by a space. Create a queue with these integers as elements.
c) Character **'p'**: Print the elements in the queue, starting with the element at the front, separated by a space.
d) Character **'m'**: Character 'm' is followed by an integer ***k*** (***k*** <= ***n***). Reverse the first ***k*** elements in the queue, leaving the other elements in the same relative order.
e) Character **'s'**: Terminate the program.

Output format:

The output (if any) of each command should be printed on a separate line.

Sample Input:

```
c 9

10 20 30 40 50 60 70 80 90

p

m 3

p

s
```

Sample Output:

```
10 20 30 40 50 60 70 80 90
30 20 10 40 50 60 70 80 90
```

4. Write a C Program to implement a Priority Queue of integers using array, in which the largest element is having the highest priority. The input should be read from the file ***input.txt*** and output should be written to the file ***output.txt***. Your program must support the following functions:

main() - Repeatedly read a character 'e', 'd', 'p' or 's' from the file and call the given functions appropriately until character 's' is entered.

enqueue(Q, element) – Insert the data specified by ***element*** into priority queue ***Q***. Return 999, if the queue is full.

dequeue(Q) – Remove and return the highest priority element from the priority queue ***Q***. If the queue is empty, return -1.

print(Q, fp) – Print the elements in the queue, starting with the element at the front, separated by a space. If the queue is empty, print -1.

Input format:

First line of the input file contains an integer value ***c***, 0 < ***c*** < 1000, which is the capacity of the queue. Subsequent lines in the input file may contain,

a) character **'e'** followed by an integer which is to be enqueued into the queue. Print 999, if the queue is full.
b) character **'d'** to dequeue the highest priority element from the queue and print it. Print -1, if the queue is empty.
c) character **'p'** to print the elements in the queue, starting with the element at the front, separated by a space. If the queue is empty, print -1.
d) character **'s'** to stop the program.

Output format:

The output (if any) of each command should be printed on a separate line.

Sample Input:

```
3
e 45
e 20
e 89
p
e 56
d
p
d
p
d
d
p
s
```

Sample Output:

```
45 20 89
999
89
45 20
45
20
20
-1
-1
```