User Id : weizhe.goh@digipen.edu | Started : 2020.06.10 09:25:54 | Submitted : 2020.06.15 02:41:49 | Score : 100%

DigiPen

Classes

Assignment

© 2020, DigiPen Institute of Technology. All Rights Reserved

Rules

Read carefully and check all rules you agree with:

- ▨ Your code must represent your own individual work. Cheating of any kind (copying someone else's work, allowing others to copy your work, collaborating, etc.) will not be tolerated and will be dealt with SEVERELY.
- ▨ Each exercise has description which must be strictly followed.
- ▨ All programs must pass all tests in the main function (when given) to get the final grade. You are not allowed to make any change in the main function in this case.
- ▨ Keep the code proper formatted (correct indentation, max line width is 40 characters).
- ▨ Every week the instructor is available **during the lab time** to discuss following matters:
  - your disagreement with rule in this card,
  - misunderstanding of the current assignment specs,
  - solution for given problems.

Part 1. Review. Specs

Fix all problems with your linked list from the previous lab. Submit the new version here not sooner than after discussion all common problems in the class. Make sure the code has:

- ▨ No globals.
- ▨ Correct namespace content, function name and parameter.
- ▨ Handle empty lists: print(nullptr) must output nothing.
- ▨ No memory leaks.
- ▨ TBD

★ Part 1. Review. Code 1 ✓

Run

```
#include <iostream>
#include <cstdlib>

namespace List
{
  struct Node
  {
    int data;
    Node *next;
  };

  void print(Node *list)
  {
    Node* current = list;

    if(list == nullptr)
    {
      return;
    }

    while(current)
    {
      if(current->next)
      {
        std::cout <<current->data <<",";
      }
      else
      {
        std::cout << current->data;
```

Part 2. Specs

- ▨ Copy **print** function from part 1. Change it to comply with following:
  - ▨ When list is empty do not output anything but the new line character (std::endl).
  - ▨ When list is not empty print all elements started from the given by address as a parameter and output the new line character after the last element.
- ▨ Develop function **push_front** that takes an address of the head and data to make new node with given data first in the list.
- ▨ Develop function **push_back** that takes an address of the head and data to make new node with given data last in the list.
- ▨ Develop function **clear** that takes an address of the head and removes all nodes from the list.

```
      }
      current = current->next;
    }
  }
}

using namespace List;

int main()
{
  Node *head = nullptr;
  head = new Node;
  Node *node1 = new Node;
  Node *node2 = new Node;

  head->data = 1;
  node1->data = 2;
  node2->data = 3;

  head->next = node1;
  node1->next = node2;
  node2->next = NULL;

  print(head);

  delete node2;
  delete node1;
  delete head;

  return 0;
}
```

```
1,2,3
```

Part 2. Code 1

Copy code from your previous lab and fix it to pass the following given test cases. Expected output is (without ")
"1
2,1
2,1,3
2,1,3,4

"

Run

```
#include <iostream>
#include <cstdlib>

namespace List
{
  struct Node
  {
    int data;
    Node *next;
  };

  void push_back(Node **list, int value)
  {
```

Survey

- What is approximate number of hours you spent implementing this assignment?

1

- Indicate the specific portions of the assignment that gave you the most trouble

Part 2

```
  Node *newNode = new Node;
  Node *current = nullptr;

  if(list == nullptr)
  {
    return;
  }

  newNode->data = value;
  newNode->next = nullptr;

  if(*list == nullptr)
  {
    *list = newNode;
    return;
  }

  current = *list;

  while(current->next)
  {
    current = current->next;
  }

  current->next = newNode;
}

void push_front(Node **list,int value)
{
  Node *newNode = new Node;

  if(list == nullptr)
  {
    return;
  }

  newNode->data = value;
  newNode->next = *list;

  *list = newNode;
}

void print(Node *list)
{
  Node* current = list;

  if(list == nullptr)
  {
    std::cout << std::endl;
    return;
  }

  while(current)
  {
    if(current->next)
    {
      std::cout <<current->data <<",";
    }
    else
    {
      std::cout << current->data;
    }
    current = current->next;
  }

  std::cout << std::endl;
  }

  void clear(Node **list)
  {
    Node* deleteNode = nullptr;
    Node* current = nullptr;

    if(list == nullptr)
    {
      return;
    }

    current = *list;

    while(current)
    {
      deleteNode = current;
      current = current->next;
      delete deleteNode;
    }

    *list = nullptr;
  }
}
```

```
int main() {
    List::Node *pHead = nullptr;
    List::push_front(&pHead, 1);
    List::print(pHead);  //1
    List::push_front(&pHead, 2);
    List::print(pHead);  //2,1
    List::push_back(&pHead, 3);
    List::print(pHead);  //2,1,3
    List::push_back(&pHead, 4);
    List::print(pHead);  //2,1,3,4
    // Remove all nodes from the list
    List::clear(&pHead);
    List::print(pHead);
    return 0;
}
```

```
1
2,1
2,1,3
2,1,3,4
```

By signing this document you fully agree that all information provided therein is complete and true in all respects.	Responder sign:

Copyright © 2020 | Powered by MyTA | www.mytaonline.com