Shiv Nadar University

CSD101: Introduction to Computing and Programming

Quiz #4 (Lab quiz)

`Max marks: 35` `27-Nov-2021`

`Time:12.00-12.55pm + 5mins (grace period for submission)`

1. *Your C program file should be uploaded to BB.*
2. *You* **must submit by 1.00pm**. *Late submissions may not be graded.*
3. *You can submit* **at most twice**. *The second submission will be graded if you have submitted twice.*
4. *Marks rubric: If the program implements the required functionality and compiles (with gcc) 5 marks; if it compiles and gives partially correct output on test data 20 marks; if it compiles and gives fully correct output on test data 35 marks.*
5. *If you have problems while submitting the quiz on BB refreshing the screen often helps.*

1. The program below implements a circular linked list. Comments in the code describe/give details of the code or declarations. You should write code at the locations where you see the comment `/* INSERT YOUR CODE HERE */`. It is in located in the function `insert` and the `main` function.

   The code starts on the next page. [35]

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   //structure for Node of a circular list
   int data;
   struct Node *next;
};

struct CList {
   /* Structure for CList.
    len - length of the circular list.
    front - pointer to first node in the list
    rear - pointer to the last node in the list
   */
   unsigned int len;
   struct Node *front, *rear;
};

int isEmpty(struct CList *clp) {
   /*Returns True if clp is empty False otherwise*/
   return (clp->front==NULL && clp->rear==NULL && clp->len==0);
}

void printCList(struct CList *clp) {
   /*Prints the circular list in square brackets.
   Prints -> at the end to indicate circularity.
   It also prints the length of the list. */
   int l=clp->len;
   struct Node *lst=clp->front;
   printf("[");
   for (int i=1; i<=l; i++) {
      printf("%d ",lst->data);
      lst=lst->next;
   }
   printf("->]    %d\n",clp->len);
   return;
}

struct CList *create() {
   /* Creates an empty CList */
   struct CList *clp=malloc(sizeof(struct CList));
   clp->len=0;
   clp->front=NULL;
   clp->rear=NULL;
   return clp;
}

struct CList *insert(struct CList *clp, int pos, int info) {
   /* Inserts a node in clp at position pos with data field
   containing value info. If pos<=1 then inserts it before front.
   If pos>length of circular list then inserts it after rear.
   if 1<pos<len of circular list then inserts it at position pos.
   Hint: Break the insert code into 4 parts: 1) clp is
   empty, 2) clp is not empty, insert before front 3) clp is not empty,
   insert after rear, 4) clp is not empty, insert in between. */

    /* INSERT YOUR CODE HERE */
```

```
}

int main(int argc, char *argv[]) {
   /* In the main function you have to read the file name from the
   command line. The file contains multiple lines where each
   line contains two integers separated by white space. The first
   integer gives the position and the second integer gives the data
   that must be added to the circular list.
   For example if the file contains the following lines:
   1 10
   -1 -2
   10 3
   1 4
   3 5
   The circular list printed at the end will be:
   [4 -2 5 10 3 ->]    5

   Print the list after every addition to the circular list.
   */
   FILE *finp;
   struct CList *clp=create();
   int pos, info;
   if (argc<2) {
      printf("Error: Need input filename argument on command line\n");
      exit(0);
   }

    /* INSERT YOUR CODE HERE */
}
```

Solution:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
   //structure for Node of a circular list
   int data;
   struct Node *next;
};

struct CList {
   /* Structure for CList.
   len - length of the circular list.
   front - pointer to first node in the list
   rear - pointer to the last node in the list
   */
   unsigned int len;
   struct Node *front, *rear;
};
```

```
int isEmpty(struct CList *clp) {
   /*Returns True if clp is empty False otherwise*/
   return (clp->front==NULL && clp->rear==NULL && clp->len==0);
}

void printCList(struct CList *clp) {
   /*Prints the circular list in square brackets.
   Prints -> at the end to indicate circularity.
   It also prints the length of the list. */
   int l=clp->len;
   struct Node *lst=clp->front;
   printf("[");
   for (int i=1; i<=l; i++) {
      printf("%d ",lst->data);
      lst=lst->next;
   }
   printf("->]    %d\n",clp->len);
   return;
}

struct CList *create() {
   /* Creates an empty CList */
   struct CList *clp=malloc(sizeof(struct CList));
   clp->len=0;
   clp->front=NULL;
   clp->rear=NULL;
   return clp;
}

struct CList *insert(struct CList *clp, int pos, int info) {
   /* Inserts a node in clp at position pos with data field
   containing value info. If pos<=1 then inserts it before front.
   If pos>length of circular list then inserts it after rear.
   if 1<pos<len of circular list then inserts it at position pos.

   Hint: Break the insert code into 4 parts: 1) clp is
   empty, 2) clp is not empty, insert before front 3) clp is not empty,
   insert after rear, 4) clp is not empty, insert in between. */

    /* INSERT YOUR CODE HERE */

   struct Node *new=malloc(sizeof(struct Node));
   new->data=info;
   new->next=NULL;
   if (isEmpty(clp)) {//clp is empty
```

```
      clp->len+=1;
      clp->front=new;
      clp->rear=new;
      new->next=clp->front;//or =new
   }
   else {//clp non-empty
      if (pos<=1) {//add before front
         clp->len++;
         new->next=clp->front;
         clp->front=new;
         (clp->rear)->next=new;
      }
      else
         if (pos>clp->len) {//add after rear
            clp->len++;
            clp->rear->next=new;
            new->next=clp->front;
            clp->rear=new;
         }
         else {//add in between
            int p=1;
            struct Node *tmp=clp->front;
            while (p++<pos-1) tmp=tmp->next;
            clp->len++;
            new->next=tmp->next;
            tmp->next=new;
         }
   }
   return clp;
}

int main(int argc, char *argv[]) {
   /* In the main function you have to read the file name from the
   command line. The file contains multiple lines where each
   line contains two integers separated by white space. The first
   integer gives the position and the second integer gives the data
   that must be added to the circular list.
   For example if the file contains the following lines:
   1 10
   -1 -2
   10 3
   1 4
   3 5
   The circular list printed at the end will be:
   [4 -2 5 10 3 ->]    5
   Print the list after every addition to the circular list.
   */
   FILE *finp;
   struct CList *clp=create();
   int pos, info;
   if (argc<2) {
      printf("Error: Need input filename argument on command line\n");
      exit(0);
   }

    /* INSERT YOUR CODE HERE */

   finp=fopen(argv[1], "r");
   if (finp==NULL ){printf("Unable to open file %s\n",argv[1]); exit(0);}
   while (fscanf(finp,"%d%d", &pos, &info)!=EOF) {
      insert(clp, pos, info);
      printCList(clp);
   }
   fclose(finp);
   printCList(clp);
   exit(0);
}
```