



CS 2263 - FR01A

# Assignment 6

By Ngoc Phuong Anh Nguyen - 3712361

November 2021

## Question 1:

Create a stack ADT module to manage Point2D data. The module should handle stack creation, push, pop, peek, reporting the stack contents and stack destruction. You should use your current Point2D module wherever possible. Test your module by implementing it with the playStack.c program.

### a. Source Code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include "Point2D.h"
5
6 Point2D* mallocPoint2D()
7 {
8     Point2D* pPtThis;
9     pPtThis = (Point2D*) malloc(sizeof(Point2D));
10    return pPtThis;
11 }
12
13 Point2D* createPoint2D(double x, double y)
14 {
15     Point2D* pPtThis;
16     pPtThis = mallocPoint2D();
17     if(pPtThis == NULL)
18     {
19         return pPtThis;
20     }
21     pPtThis->x = x;
22     pPtThis->y = y;
23     return pPtThis;
24 }
25
26 void freePoint2D(Point2D* pThis)
27 {
28     free(pThis);
29 }
30
31 double getXPoint2D(Point2D* pThis)
32 {
33     return pThis->x;
34 }
35
36 double getYPoint2D(Point2D* pThis)
37 {
38     return pThis->y;
39 }
40
41 void setPoint2D(Point2D* pPt, double x, double y)
42 {
43     pPt->x = x;
44     pPt->y = y;
45     return;
46 }
47
48 double getDistancePoint2D(Point2D* pThis, Point2D* pThat)
49 {
50     return sqrt(pow((getXPoint2D(pThis) - getXPoint2D(pThat)), 2)
51               + pow((getYPoint2D(pThis) - getYPoint2D(pThat)), 2));
52 }
```

Figure 1: Source Code of Point2D.c

```
1 #ifndef POINT2D_H
2 #define POINT2D_H
3
4 typedef struct point2D
5 {
6     double x;
7     double y;
8 }
9 Point2D;
10
11 typedef struct pt2link
12 {
13     Point2D* payload;
14     struct pt2link* next;
15 }
16 PtLink, *pPtLink;
17
18 typedef struct pointList
19 {
20     pPtLink head;
21     pPtLink tail;
22     pPtLink iterator; /* not implemented */
23     int nLinks;
24 }PointList;
25
26 Point2D* mallocPoint2D();
27 Point2D* createPoint2D(double x, double y);
28 void freePoint2D(Point2D* pThis);
29 double getXPoint2D(Point2D* pThis);
30 double getYPoint2D(Point2D* pThis);
31 void setPoint2D(Point2D* pPt, double x, double y);
32 double getDistancePoint2D(Point2D* pThis, Point2D* pThat);
33
34 #endif
```

Figure 2: Source Code of Point2D.h

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "Point2D.h"
4 #include "stack.h"
5 // Other supported functions
6 pPtLink createPointLink(Point2D* point)
7 {
8     pPtLink pPointLink = (pPtLink) malloc(sizeof(PtLink));
9     if(pPointLink == (pPtLink)NULL)
10     {
11         return pPointLink;
12     }
13     pPointLink->payload = point;
14     return pPointLink;
15 }
16
17 void freePointLink(pPtLink pPointLink)
18 {
19     free(pPointLink);
20     return;
21 }
22
23 // stack creation
24 PointList* mallocPointList()
25 {
26     PointList* pointListThis = (PointList*) malloc(sizeof(PointList));
27     if(pointListThis == (PointList*)NULL )
28     {
29         return pointListThis;
30     }
31     pointListThis->head = (pPtLink)NULL;
32     pointListThis->tail = (pPtLink)NULL;
33     pointListThis->iterator = (pPtLink)NULL;
34     pointListThis->nLinks = 0;
35
36     return pointListThis;
37 }
38
39 // push
40 int push(Point2D* newPoint, PointList* pointList)
41 {
42     pPtLink pPointLink = createPointLink(newPoint);
43     if(pPointLink == (pPtLink)NULL)
44     {
45         return 1;
46     }
47     if(pointList->head == pointList->tail && pointList->head == NULL)
48     {
49         pointList->head = pPointLink;
50         pointList->tail = pointList->head;
51     }
52     else
53     {
54         pPointLink->next = pointList->head;
55         pointList->head = pPointLink;
56     }
57     pointList->nLinks++;
58     return 0; //success
59 }
60

```

Figure 3: Source Code of stack.c (1)

```

61 // pop
62 int pop(PointList* pointList)
63 {
64     pPtLink temp;
65     if(pointList->head == (pPtLink)NULL)
66     {
67         return EOF;
68     }
69     temp = pointList->head;
70     pointList->head = pointList->head->next;
71     pointList->nLinks--;
72     printf("Popped point: (%.1lf, %.1lf)\n", getXPoint2D(temp->payload), getYPoint2D(temp->payload));
73     return 0; //success
74 }
75
76 // peek
77 void peek(PointList* pointList)
78 {
79     if(pointList->head == (pPtLink)NULL)
80     {
81         printf("The list is empty, unable to peek the point\n");
82     }
83     else
84     {
85         Point2D* point = pointList->head->payload;
86         printf("Peeked point: (%.1lf, %.1lf)\n", getXPoint2D(point), getYPoint2D(point));
87     }
88 }
89
90 // reporting the stack contents
91 void print(PointList* pointList)
92 {
93     pPtLink current = pointList->head;
94     int i = 0;
95     if(current == (pPtLink)NULL)
96     {
97         printf("The list is empty\n");
98         return;
99     }
100     while(current != (pPtLink)NULL)
101     {
102         printf("Point %d: (%.1lf, %.1lf)\n", i, current->payload->x, getYPoint2D(current->payload));
103         printf("- PtLink: %p\n- payload: %p\n- next: %p\n\n", current, current->payload, current->next);
104         i++;
105         current = current->next;
106     }
107 }
108
109 // stack destruction
110 void freePointList(PointList* pointList)
111 {
112     pPtLink current = pointList->head;
113     while(pointList->head != (pPtLink)NULL)
114     {
115         current = pointList->head->next;
116         freePointLink(pointList->head);
117         pointList->head = current;
118     }
119     free(pointList);
120     return;
121 }
122

```

Figure 4: Source Code of stack.c (2)

```

1 #ifndef STACK_H
2 #define STACK_H
3 #include "Point2D.h"
4
5 pPtLink createPointLink(Point2D* point);
6 void freePointLink(pPtLink pPointLink);
7 PointList* mallocPointList();
8 int push(Point2D* newPoint, PointList* pointList);
9 int pop(PointList* pointList);
10 void peek(PointList* pointList);
11 void print(PointList* pointList);
12 void freePointList(PointList* pointList);
13
14 #endif

```

Figure 5: Source Code of stack.h



```

1 // playStack.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include "Point2D.h"
5 #include "stack.h"
6 #define PUSH 1
7 #define POP 0
8 #define LIST 2
9 #define PEEK 3
10 int main(int argc, char* argv[])
11 {
12     int iChoice, iNRead, failure;
13     Point2D* point;
14     double x,y;
15     Point2D* newPoint;
16     PointList* pointList = mallocPointList();
17
18     printf("Choice (1=add, 0=remove, 2=list, 3=peek): ");
19     iNRead = scanf("%d", &iChoice);
20     while(iNRead == 1)
21     {
22         switch(iChoice)
23         {
24             case PUSH:
25                 printf("Point value to add: ");
26                 int check = scanf("%lf%lf", &x,&y);
27                 while(check != 2)
28                 {
29                     printf("Please input the point value again: ");
30                     check = scanf("%lf%lf", &x,&y);
31                 }
32                 newPoint = createPoint2D(x,y);
33                 failure = push(newPoint, pointList);
34                 if(failure)
35                 {
36                     printf("Unable to add the value to the list.\n");
37                 }
38                 break;
39             case POP:
40                 failure = pop(pointList);
41                 if(failure != 0)
42                 {
43                     printf("Unable to pop the value from the list.\n");
44                 }
45                 break;
46             case LIST:
47                 print(pointList);
48                 break;
49             case PEEK:
50                 peek(pointList);
51                 break;
52             default:
53                 freePoint2D(newPoint);
54                 freePointList(pointList);
55                 return 0;
56         }
57         printf("Choice (1=add, 0=remove, 2=list, 3=peek): ");
58         iNRead = scanf("%d", &iChoice);
59     }
60 }

```

Figure 6: Source Code of playStack.c

## b. Output:

<pre> [anguyen5@gcl12m30 Assignment 6]\$ make playStack gcc -c stack.c gcc -c Point2D.c gcc -lm stack.o Point2D.o playStack.c -o playStack [anguyen5@gcl12m30 Assignment 6]\$ ./playStack Choice (1=add, 0=remove, 2=list, 3=peek): 1 Point value to add: 2.5 3.5 Choice (1=add, 0=remove, 2=list, 3=peek): 2 Point 0: (2.5, 3.5) - PtLink: 0x1f8d060 - payload: 0x1f8d040 - next: (nil)  Choice (1=add, 0=remove, 2=list, 3=peek): 1 Point value to add: 5.6 8.8 Choice (1=add, 0=remove, 2=list, 3=peek): 2 Point 0: (5.6, 8.8) - PtLink: 0x1f8d0a0 - payload: 0x1f8d080 - next: 0x1f8d060  Point 1: (2.5, 3.5) - PtLink: 0x1f8d060 - payload: 0x1f8d040 - next: (nil)  Choice (1=add, 0=remove, 2=list, 3=peek): 3 Peeked point: (5.6, 8.8) Choice (1=add, 0=remove, 2=list, 3=peek): 0 </pre>	<pre> Popped point: (5.6, 8.8) Choice (1=add, 0=remove, 2=list, 3=peek): 2 Point 0: (2.5, 3.5) - PtLink: 0x1f8d060 - payload: 0x1f8d040 - next: (nil)  Choice (1=add, 0=remove, 2=list, 3=peek): 0 Popped point: (2.5, 3.5) Choice (1=add, 0=remove, 2=list, 3=peek): 2 The list is empty Choice (1=add, 0=remove, 2=list, 3=peek): 0 Unable to pop the value from the list. Choice (1=add, 0=remove, 2=list, 3=peek): 1 Point value to add: 5.5 4 Choice (1=add, 0=remove, 2=list, 3=peek): 2 Point 0: (5.5, 4.0) - PtLink: 0x1f8d0e0 - payload: 0x1f8d0c0 - next: (nil)  Choice (1=add, 0=remove, 2=list, 3=peek): 0 Popped point: (5.5, 4.0) Choice (1=add, 0=remove, 2=list, 3=peek): 2 The list is empty Choice (1=add, 0=remove, 2=list, 3=peek): 3 The list is empty, unable to peek the point Choice (1=add, 0=remove, 2=list, 3=peek): u [anguyen5@gcl12m30 Assignment 6]\$ </pre>
---	---

Figure 7: Output of playStack.c

## Question 2:

Create a queue ADT module to manage Point2D data. The module should handle queue creation, enqueueing, dequeuing, peek (look at the next value to be dequeued), reporting the queue contents and queue destruction. You should use your current Point2D module wherever possible. Test your module by implementing it with the playQueue.c program.

### a. Source Code:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "Point2D.h"
4  #include "queue.h"
5  // Other supported functions
6  pPtLink createPointLink(Point2D* point)
7  {
8      pPtLink pPointLink = (pPtLink) malloc(sizeof(PtLink));
9      if(pPointLink == (pPtLink)NULL)
10     {
11         return pPointLink;
12     }
13     pPointLink->payload = point;
14     return pPointLink;
15 }
16
17 void freePointLink(pPtLink pPointLink)
18 {
19     free(pPointLink);
20     return;
21 }
22
23 // queue creation
24 PointList* mallocPointList()
25 {
26     PointList* pointListThis = (PointList*) malloc(sizeof(PointList));
27     if(pointListThis == (PointList*)NULL )
28     {
29         return pointListThis;
30     }
31     pointListThis->head = (pPtLink)NULL;
32     pointListThis->tail = (pPtLink)NULL;
33     pointListThis->iterator = (pPtLink)NULL;
34     pointListThis->nLinks = 0;
35     return pointListThis;
36 }
37
38 // enqueue
39 int enqueue(Point2D* newPoint, PointList* pointList)
40 {
41     pPtLink pPointLink = createPointLink(newPoint);
42     if(pPointLink == (pPtLink)NULL)
43     {
44         return 1;
45     }
46     if(pointList->nLinks)
47     {
48         pointList->tail->next = pPointLink;
49     }
50     else
51     {
52         pointList->head = pPointLink;
53     }
54     pointList->tail = pPointLink;
55     pointList->nLinks++;
56     return 0; //success
57 }
58
59 // dequeuing
60 int dequeue(PointList* pointList)
61 {
62     if(!pointList->nLinks)
```

Figure 8: Source Code of queue.c (1)

```
63     return EOF;
64 }
65 pPtLink temp = pointList->head;
66 Point2D* tempPoint = temp->payload;
67 pointList->head = temp->next;
68 printf("Dequeued point: (%.1lf, %.1lf)\n", getXPoint2D(tempPoint), getYPoint2D(tempPoint));
69 free(temp);
70 pointList->nLinks--;
71 return 0; //success
72 }
73
74 // peek (look at the next value to be dequeued)
75 void peek(PointList* pointList)
76 {
77     if(pointList->head == (pPtLink)NULL)
78     {
79         printf("The list is empty, unable to peek the point\n");
80     }
81     else
82     {
83         Point2D* point = pointList->head->payload;
84         printf("Peeked point: (%.1lf, %.1lf)\n", getXPoint2D(point), getYPoint2D(point));
85     }
86 }
87
88 // reporting the queue contents
89 void print(PointList* pointList)
90 {
91     pPtLink current = pointList->head;
92     int i = 0;
93     if(i == pointList->nLinks)
94     {
95         printf("The list is empty\n");
96         return;
97     }
98     while(i < pointList->nLinks)
99     {
100         printf("Point %d: (%.1lf, %.1lf)\n", i, getXPoint2D(current->payload), getYPoint2D(current->payload));
101         printf("Address:\n");
102         printf("- PtLink: %p\n", current);
103         printf("- payload: %p\n", current->payload);
104         printf("- next: %p\n\n", current->next);
105         i++;
106         current = current->next;
107     }
108 }
109
110 // and queue destruction
111 void freePointList(PointList* pointList)
112 {
113     pPtLink current = pointList->head;
114     while(pointList->head != (pPtLink)NULL)
115     {
116         current = pointList->head->next;
117         freePointLink(pointList->head);
118         pointList->head = current;
119     }
120     free(pointList);
121     return;
122 }
```

Figure 9: Source Code of queue.c (2)

```

1  #ifndef QUEUE_H
2  #define QUEUE_H
3  #include "Point2D.h"
4
5  pPtLink createPointLink(Point2D* point);
6  void freePointLink(pPtLink pPointLink);
7  PointList* mallocPointList();
8  int enqueue(Point2D* newPoint, PointList* pointList);
9  int dequeue(PointList* pointList);
10 void peek(PointList* pointList);
11 void print(PointList* pointList);
12 void freePointList(PointList* pointList);
13
14 #endif

```

Figure 10: Source Code of queue.h

```

1  // playQueue.c
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include "Point2D.h"
5  #include "queue.h"
6  #define ENQUEUE 1
7  #define DEQUEUE 0
8  #define LIST 2
9  #define PEEK 3
10 int main(int argc, char ** argv)
11 {
12     int iChoice, iNRead, failure;
13     double x,y;
14     Point2D* newPoint;
15     PointList* pointList = mallocPointList();
16
17     /* Processing loop */
18     printf("Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): ");
19     iNRead = scanf("%d", &iChoice);
20     while(iNRead == 1)
21     {
22         switch(iChoice)
23         {
24             case ENQUEUE:
25                 printf("Point value to add: ");
26                 int check = scanf("%lf%lf", &x,&y);
27                 while(check != 2)
28                 {
29                     printf("Please input the point value again: ");
30                     check = scanf("%lf%lf", &x,&y);
31                 }
32                 newPoint = createPoint2D(x,y);
33                 failure = enqueue(newPoint, pointList);
34                 if(failure)
35                 {
36                     printf("Unable to add the value to the list.\n");
37                 }
38                 break;
39             case DEQUEUE:
40                 failure = dequeue(pointList);
41                 if(failure != 0)
42                 {
43                     printf("Unable to dequeue the value from the list.\n");
44                 }
45                 break;
46             case LIST:
47                 print(pointList);
48                 break;
49             case PEEK:
50                 peek(pointList);
51                 break;
52             default:
53                 freePoint2D(newPoint);
54                 freePointList(pointList);
55                 return 0;
56         }
57         printf("Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): ");
58         iNRead = scanf("%d", &iChoice);
59     }
60     return EXIT_SUCCESS;
61 }

```

Figure 11: Source Code of playQueue.c

## b. Output:

```
[anguyen5@gcl12m30 Assignment 6]$ make playQueue
gcc -c queue.c
gcc -c Point2D.c
gcc -lm queue.o Point2D.o playQueue.c -o playQueue
[anguyen5@gcl12m30 Assignment 6]$ ./playQueue
Point value to add: 4 5
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 1
Point value to add: 9.9 6
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 3
Peeked point: (4.0, 55.0)
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 2
Point 0: (4.0, 55.0)
Address:
- PtLink: 0x247a060
- payload: 0x247a0a0
- next: 0x247a0e0

Point 1: (9.9, 6.0)
Address:
- PtLink: 0x247a0e0
- payload: 0x247a0c0
- next: (nil)

Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 0
Dequeued point: (4.0, 55.0)
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 2
Point 0: (9.9, 6.0)
Address:
- PtLink: 0x247a0e0
- payload: 0x247a0c0
- next: (nil)

Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 0
Dequeued point: (9.9, 6.0)
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 3
The list is empty, unable to peek the point
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): 2
The list is empty
Choice (1=enqueue, 0=dequeue, 2=list, 3=peek): u
[anguyen5@gcl12m30 Assignment 6]$
```

Figure 12: Output of playQueue.c