

```

1
2 /* =====
3 ** #1. A C++ program of implementation of Stack.
4 ** =====*/
5
6 #include <iostream>
7 using namespace std;
8
9 struct Node
10 {
11     int data;
12     struct Node *next;
13 };
14 struct Node* top = NULL;
15
16 void push(int val)
17 {
18     struct Node* newnode = (struct Node*)malloc(sizeof(struct Node));
19     newnode->data = val;
20     newnode->next = top;
21     top = newnode;
22 }
23
24 void pop()
25 {
26     if(top==NULL)
27         cout<<"Stack Underflow"<<endl;
28     else
29     {
30         cout<<"The popped element is "<< top->data <<endl;
31         top = top->next;
32     }
33 }
34
35 void display()
36 {
37     struct Node* ptr;
38     if(top==NULL)
39         cout<<"stack is empty";
40     else
41     {
42         ptr = top;
43         cout<<"Stack elements are: ";
44         while (ptr != NULL)
45         {
46             cout<< ptr->data <<" ";
47             ptr = ptr->next;
48         }
49     }
50     cout<<endl;
51 }
52
53 int main()
54 {
55     int ch, val;
56     cout<<"1) Push in stack"<<endl;
57     cout<<"2) Pop from stack"<<endl;

```

```

58     cout<<"3) Display stack"<<endl;
59     cout<<"4) Exit"<<endl;
60     do
61     {
62         cout<<"Enter choice: "<<endl;
63         cin>>ch;
64         switch(ch)
65         {
66             case 1:
67             {
68                 cout<<"Enter value to be pushed:"<<endl;
69                 cin>>val;
70                 push(val);
71                 break;
72             }
73             case 2:
74             {
75                 pop();
76                 break;
77             }
78             case 3:
79             {
80                 display();
81                 break;
82             }
83             case 4:
84             {
85                 cout<<"Exit"<<endl;
86                 break;
87             }
88             default:
89             {
90                 cout<<"Invalid Choice"<<endl;
91             }
92         }
93     }while(ch!=4);
94     return 0;
95 }
96
97
98 /* =====
99 ** #2. A C++ program of implementation of Queue.
100 ** =====*/
101 #include <iostream>
102 using namespace std;
103 struct node {
104     int data;
105     struct node *next;
106 };
107
108 struct node* front = NULL;
109 struct node* rear = NULL;
110 struct node* temp;
111
112 void push() {
113     int val;
114     cout<<"push the element in queue : "<<endl;

```

```

115     cin>>val;
116     if (rear == NULL) {
117         rear = (struct node *)malloc(sizeof(struct node));
118         rear->next = NULL;
119         rear->data = val;
120         front = rear;
121     } else {
122         temp=(struct node *)malloc(sizeof(struct node));
123         rear->next = temp;
124         temp->data = val;
125         temp->next = NULL;
126         rear = temp;
127     }
128 }
129
130 void pop() {
131     temp = front;
132     if (front == NULL) {
133         cout<<"Underflow"<<endl;
134         return;
135     }
136     else
137     if (temp->next != NULL) {
138         temp = temp->next;
139         cout<<"Element popped from queue is : "<<front->data<<endl;
140         free(front);
141         front = temp;
142     } else {
143         cout<<"Element popped from queue is : "<<front->data<<endl;
144         free(front);
145         front = NULL;
146         rear = NULL;
147     }
148 }
149
150 void Display() {
151     temp = front;
152     if ((front == NULL) && (rear == NULL)) {
153         cout<<"Queue is empty"<<endl;
154         return;
155     }
156     cout<<"Queue elements are: ";
157     while (temp != NULL) {
158         cout<<temp->data<<" ";
159         temp = temp->next;
160     }
161     cout<<endl;
162 }
163
164 int main() {
165     int ch;
166     cout<<"1) push element to queue"<<endl;
167     cout<<"2) pop element from queue"<<endl;
168     cout<<"3) Display all the elements of queue"<<endl;
169     cout<<"4) Exit"<<endl;
170     do {
171         cout<<"Enter your choice : "<<endl;

```

```

172     cin>>ch;
173     switch (ch) {
174         case 1: push();
175         break;
176         case 2: pop();
177         break;
178         case 3: Display();
179         break;
180         case 4: cout<<"Exit"<<endl;
181         break;
182         default: cout<<"Invalid choice"<<endl;
183     }
184 } while(ch!=4);
185 return 0;
186 }
187
188
189 /* =====
190  * #3. A C++ Program to Implement Stack Using Queue
191  * =====*/
192
193 #include<stdio.h>
194 #include<iostream>
195 #include<conio.h>
196 using namespace std;
197
198 struct queue1
199 {
200     queue1 *next1;
201     int data1;
202 }*front1 = NULL, *rear1 = NULL, *q1 = NULL, *p1 = NULL, *np1 = NULL;
203
204 struct queue2
205 {
206     queue2 *next2;
207     int data2;
208 }*front2 = NULL, *rear2 = NULL, *q2 = NULL, *p2 = NULL, *np2 = NULL;
209
210 void enqueue1(int x)
211 {
212     np1 = new queue1;
213     np1->data1 = x;
214     np1->next1 = NULL;
215     if (front1 == NULL)
216     {
217         rear1 = np1;
218         rear1->next1 = NULL;
219         front1 = rear1;
220     }
221     else
222     {
223         rear1->next1 = np1;
224         rear1 = np1;
225         rear1->next1 = NULL;
226     }
227 }
228

```

```

229 int dequeue1()
230 {
231     int x;
232     if (front1 == NULL)
233     {
234         cout<<"no elements present in queue\n";
235     }
236     else
237     {
238         q1 = front1;
239         front1 = front1->next1;
240         x = q1->data1;
241         delete(q1);
242         return x;
243     }
244 }
245
246 void enqueue2(int x)
247 {
248     np2 = new queue2;
249     np2->data2 = x;
250     np2->next2 = NULL;
251     if (front2 == NULL)
252     {
253         rear2 = np2;
254         rear2->next2 = NULL;
255         front2=rear2;
256     }
257     else
258     {
259         rear2->next2 = np2;
260         rear2 = np2;
261         rear2->next2 = NULL;
262     }
263 }
264
265 int dequeue2()
266 {
267     int x;
268     if (front2 == NULL)
269     {
270         cout<<"no elements present in queue\n";
271     }
272     else
273     {
274         q2 = front2;
275         front2 = front2->next2;
276         x = q2->data2;
277         delete(q2);
278         return x;
279     }
280 }
281
282 int main()
283 {
284     int n, x, i = 0;
285     cout<<"Enter the number of elements to be entered into stack\n";

```

```

286     cin>>n;
287     while (i < n)
288     {
289         cout<<"enter the element to be entered\n";
290         cin>>x;
291         enqueue1(x);
292         i++;
293     }
294     cout<<"\n\nElements popped\n\n";
295     while (front1 != NULL || front2 != NULL)
296     {
297         if (front2 == NULL)
298         {
299             while (front1->next1 != NULL)
300             {
301                 enqueue2(dequeue1());
302             }
303             cout<<dequeue1()<<endl;
304         }
305         else if (front1 == NULL)
306         {
307             while (front2->next2 != NULL)
308             {
309                 enqueue1(dequeue2());
310             }
311             cout<<dequeue2()<<endl;
312         }
313     }
314     getch();
315 }
316
317 /* =====
318 * #4. A C++ Program to Implement Queue Using Stack
319 * =====*/
320
321 #include <stdio.h>
322 #include <stdlib.h>
323 #include <iostream>
324
325 using namespace std;
326
327 struct sNode
328 {
329     int data;
330     struct sNode *next;
331 };
332
333 void push(struct sNode **top_ref, int new_data);
334
335 int pop(struct sNode **top_ref);
336
337 struct queue
338 {
339     struct sNode *stack1;
340     struct sNode *stack2;
341 };
342

```

```

343 void enqueue(struct queue *q, int x)
344 {
345     push(&q->stack1, x);
346 }
347
348 int dequeue(struct queue *q)
349 {
350     int x;
351
352     if (q->stack1 == NULL && q->stack2 == NULL)
353     {
354         cout << "Queue is empty";
355         exit(0);
356     }
357     if (q->stack2 == NULL)
358     {
359         while (q->stack1 != NULL)
360         {
361             x = pop(&q->stack1);
362             push(&q->stack2, x);
363         }
364     }
365
366     x = pop(&q->stack2);
367     return x;
368 }
369
370 void push(struct sNode **top_ref, int new_data)
371 {
372
373     struct sNode *new_node = (struct sNode *)malloc(sizeof(struct sNode));
374
375     if (new_node == NULL)
376     {
377         cout << "Stack overflow \n";
378         exit(0);
379     }
380
381
382     new_node->data = new_data;
383
384     new_node->next = (*top_ref);
385
386     (*top_ref) = new_node;
387 }
388
389 int pop(struct sNode **top_ref)
390 {
391     int res;
392     struct sNode *top;
393
394     if (*top_ref == NULL)
395     {
396         cout << "Stack overflow \n";
397         exit(0);
398     }
399     else

```

```

400     {
401         top = *top_ref;
402         res = top->data;
403         *top_ref = top->next;
404         free(top);
405         return res;
406     }
407 }
408
409 int main()
410 {
411
412     struct queue *q = (struct queue *)malloc(sizeof(struct queue));
413     q->stack1 = NULL;
414     q->stack2 = NULL;
415     cout << "Enqueuing...";
416     cout << endl;
417     enqueue(q, 1);
418     cout << "Enqueuing...";
419     cout << endl;
420     enqueue(q, 2);
421     cout << "Enqueuing...";
422     cout << endl;
423     enqueue(q, 3);
424
425     cout << "Dequeuing...";
426     cout << dequeue(q) << " ";
427     cout << endl;
428     cout << "Dequeuing...";
429     cout << dequeue(q) << " ";
430     cout << endl;
431     cout << "Dequeuing...";
432     cout << dequeue(q) << " ";
433     cout << endl;
434 }
435
436
437 /* =====
438 * #5. A C++ program of Implementation of Linked List
439 * =====*/
440
441 #include <iostream>
442 using namespace std;
443
444 struct Node
445 {
446     int data;
447     Node *next;
448 };
449
450 class LinkedList
451 {
452
453     Node *head;
454
455 public:
456

```



```

457     LinkedList()
458     {
459         head = NULL;
460     }
461
462
463     void insert(int val)
464     {
465         Node *new_node = new Node;
466         new_node->data = val;
467         new_node->next = NULL;
468
469         if (head == NULL)
470             head = new_node;
471         else
472         {
473             new_node->next = head;
474             head = new_node;
475         }
476     }
477
478     bool search(int val)
479     {
480         Node *temp = head;
481         while (temp != NULL)
482         {
483             if (temp->data == val)
484                 return true;
485             temp = temp->next;
486         }
487         return false;
488     }
489
490     void remove(int val)
491     {
492         if (head->data == val)
493         {
494             delete head;
495             head = head->next;
496             return;
497         }
498
499         if (head->next == NULL)
500         {
501             if (head->data == val)
502             {
503                 delete head;
504                 head = NULL;
505                 return;
506             }
507             cout << "Value not found!" << endl;
508             return;
509         }
510
511         Node *temp = head;
512         while (temp->next != NULL)
513         {

```

```

514         if (temp->next->data == val)
515         {
516             Node *temp_ptr = temp->next->next;
517             delete temp->next;
518             temp->next = temp_ptr;
519             return;
520         }
521         temp = temp->next;
522     }
523
524     cout << "Value not found" << endl;
525 }
526
527 void display()
528 {
529     Node *temp = head;
530     while (temp != NULL)
531     {
532         cout << temp->data << " ";
533         temp = temp->next;
534     }
535     cout << endl;
536 }
537 };
538
539 int main()
540 {
541
542     LinkedList l;
543     // inserting elements
544     l.insert(6);
545     l.insert(9);
546     l.insert(1);
547     l.insert(3);
548     l.insert(7);
549     cout << "Current Linked List: ";
550     l.display();
551
552     cout << "Deleting 1: ";
553     l.remove(1);
554     l.display();
555
556     cout << "Deleting 13: ";
557     l.remove(13);
558
559     cout << "Searching for 7: ";
560     cout << l.search(7) << endl;
561
562     cout << "Searching for 13: ";
563     cout << l.search(13) << endl;
564 }
565
566
567 /* =====
568 * #6. A C++ program of Reverse Linked List
569 * =====*/
570

```

```

571 #include <iostream>
572 using namespace std;
573 struct Node
574 {
575     int data;
576     struct Node *next;
577     Node(int data)
578     {
579         this->data = data;
580         next = NULL;
581     }
582 };
583 struct LinkedList
584 {
585     Node *head;
586     LinkedList()
587     {
588         head = NULL;
589     }
590     void reverse()
591     {
592         Node *current = head;
593         Node *prev = NULL, *next = NULL;
594         while (current != NULL)
595         {
596             next = current->next;
597             current->next = prev;
598             prev = current;
599             current = next;
600         }
601         head = prev;
602     }
603     void print()
604     {
605         struct Node *temp = head;
606         while (temp != NULL)
607         {
608             cout << temp->data << " ";
609             temp = temp->next;
610         }
611     }
612     void push(int data)
613     {
614         Node *temp = new Node(data);
615         temp->next = head;
616         head = temp;
617     }
618 };
619 int main()
620 {
621     LinkedList ll;
622     ll.push(40);
623     ll.push(30);
624     ll.push(20);
625     ll.push(10);
626     cout << "old linked list\n";
627     ll.print();

```

```

628     ll.reverse();
629     cout << "\nnew Linked list \n";
630     ll.print();
631     return 0;
632 }
633
634 /* =====
635  * #7. A C++ program of Implementation of Singly Linked List.
636  * =====*/
637
638
639 #include <iostream>
640 using namespace std;
641
642
643 struct Node
644 {
645     int data;
646     Node *next;
647 };
648
649 class LinkedList
650 {
651     Node *head;
652
653 public:
654     LinkedList()
655     {
656         head = NULL;
657     }
658
659     void insert(int val)
660     {
661         Node *new_node = new Node;
662         new_node->data = val;
663         new_node->next = NULL;
664
665         if (head == NULL)
666             head = new_node;
667         else
668         {
669             new_node->next = head;
670             head = new_node;
671         }
672     }
673
674     bool search(int val)
675     {
676         Node *temp = head;
677         while (temp != NULL)
678         {
679             if (temp->data == val)
680                 return true;
681             temp = temp->next;
682         }
683         return false;
684     }

```

```

685
686 void remove(int val)
687 {
688     if (head->data == val)
689     {
690         delete head;
691         head = head->next;
692         return;
693     }
694
695     if (head->next == NULL)
696     {
697         if (head->data == val)
698         {
699             delete head;
700             head = NULL;
701             return;
702         }
703
704         cout << "Value not found!" << endl;
705         return;
706     }
707
708     Node *temp = head;
709     while (temp->next != NULL)
710     {
711         if (temp->next->data == val)
712         {
713             Node *temp_ptr = temp->next->next;
714             delete temp->next;
715             temp->next = temp_ptr;
716             return;
717         }
718         temp = temp->next;
719     }
720
721     cout << "Value not found" << endl;
722 }
723
724 void display()
725 {
726     Node *temp = head;
727     while (temp != NULL)
728     {
729         cout << temp->data << " ";
730         temp = temp->next;
731     }
732     cout << endl;
733 }
734 };
735
736 int main()
737 {
738
739     LinkedList l;
740     // inserting elements
741     l.insert(6);

```

```

742     l.insert(9);
743     l.insert(1);
744     l.insert(3);
745     l.insert(7);
746     cout << "Current Linked List: ";
747     l.display();
748
749     cout << "Deleting 1: ";
750     l.remove(1);
751     l.display();
752
753     cout << "Deleting 13: ";
754     l.remove(13);
755
756     cout << "Searching for 7: ";
757     cout << l.search(7) << endl;
758
759     cout << "Searching for 13: ";
760     cout << l.search(13) << endl;
761 }
762
763
764 /* =====
765  * #8. A C++ program of Implementation of Doubly Linked List.
766  * =====*/
767
768 #include <iostream>
769 #include "stdio.h"
770 #include "conio.h"
771
772 using namespace std;
773 int insertdata(int x);
774 void display();
775 void deleteint(int x);
776 void reversel();
777 int searchint(int x);
778 int compare_fn(int a, int b)
779 {
780     if (a > b)
781         return 1;
782     else if (b > a)
783         return -1;
784 }
785 int compare_no = 1;
786 struct node
787 {
788     int data;
789     node *prev;
790     node *next;
791 };
792 node *top = NULL;
793 int main()
794 {
795     int ch, d, y;
796     char ans = 'y';
797     while (ans == 'y')
798     {

```

```

799     cout << "\n\t 1.Insert          2. Delete          3.Reverse          4.EXIT\nEnter Choice : ";
800     cin >> ch;
801     if (ch == 1)
802     {
803         cout << "Enter An Element To be inserted : ";
804         cin >> d;
805         d = insertdata(d);
806         display();
807     }
808     else if (ch == 2)
809     {
810         cout << "Enter Element To Be Deleted : ";
811         cin >> d;
812         deleteint(d);
813         display();
814     }
815     else if (ch == 3)
816         reversel();
817     else
818         return 0;
819 }
820 return 0;
821 }
822 int searchint(int x)
823 {
824     int count = 0;
825     node *searchele = top;
826     while (searchele != NULL)
827     {
828         if (compare_fn(x, searchele->data) == compare_no)
829         {
830             searchele = searchele->next;
831             count += 1;
832         }
833         else
834             break;
835     }
836     return count;
837 }
838 int insertdata(int x)
839 {
840     if (top == NULL)
841     {
842         top = new node;
843         top->data = x;
844         top->next = NULL;
845         top->prev = NULL;
846     }
847     else if (compare_fn(top->data, x) == compare_no)
848     {
849         node *n = new node;
850         n->data = x;
851         n->next = top;
852         n->prev = NULL;
853         top->prev = n;
854         top = n;

```

```

855     }
856     else
857     {
858         int c = searchint(x);
859         node *insertele = top;
860         for (int i = 0; i < c - 1; i++)
861             insertele = insertele->next;
862         node *n = new node;
863         n->data = x;
864         node *b = insertele->next;
865         node *N = insertele;
866         n->prev = insertele;
867         n->next = b;
868         insertele->next = n;
869         if (b != NULL)
870             b->prev = n;
871     }
872 }
873 void display()
874 {
875     cout << "Element In The Linked List Are : ";
876     node *disp = top;
877     while (disp != NULL)
878     {
879         cout << " " << disp->data;
880         if (disp->next == NULL)
881         {
882             break;
883         }
884         disp = disp->next;
885     }
886 }
887 void deleteint(int x)
888 {
889     node *del = top;
890     if (del->data == x)
891     {
892         if (del->next == NULL && del->prev == NULL)
893         {
894             top = NULL;
895             return;
896         }
897         del->next->prev = NULL;
898         top = del->next;
899     }
900     else
901     {
902         node *delsuc = del->next;
903         if (del == NULL)
904         {
905             cout << "\nElement Not Found\n";
906             return;
907         }
908         if (delsuc == NULL)
909         {
910             cout << "\nElement Not Found\n";
911             return;

```



```

912     }
913     while (delsuc->data != x)
914     {
915         del = del->next;
916         delsuc = delsuc->next;
917         if (del == NULL)
918         {
919             cout << "\nElement Not Found\n";
920             return;
921         }
922         if (delsuc == NULL)
923         {
924             cout << "\nElement Not Found\n";
925             return;
926         }
927     }
928     del->next = delsuc->next;
929     if (delsuc->next != NULL)
930         delsuc->next->prev = del;
931 }
932 }
933 void reversel()
934 {
935     node *a = top;
936     node *b, *c, *d;
937     while (a != NULL)
938     {
939         d = a;
940         c = a->next;
941         b = a->prev;
942         a->prev = a->next;
943         a->next = b;
944         a = c;
945     }
946     top = d;
947     cout << "After Reversing the linked list";
948     display();
949     compare_no *= -1;
950 }

```