

```

33  /* *** Codifique aqui a funcao bubbleSort versão 1 *** */
34  void bubbleSort (int *vetor, int N) {
35      int i, aux, troca = 1;
36      while (troca==1) {
37          troca = 0;
38          for (i = 0; i < N-1; i++) {
39              if (vetor[i] > vetor[i+1]) {
40                  aux = vetor[i];
41                  vetor[i] = vetor[i+1];
42                  vetor[i+1] = aux;
43                  troca = 1;
44              }
45          }
46      }
47  }

```

```

50  /* *** Codifique aqui a funcao bubbleSort versão 2 *** */
51  void bubbleSort2 (int *vetor, int N) {
52      int i, j, aux, troca = 1;
53      for (i = 0; (i < N-1) && troca; i++) {
54          troca = 0;
55          for (j = 0; j < N-i-1; j++) {
56              if (vetor[j] > vetor[j+1]) {
57                  aux = vetor[j];
58                  vetor[j] = vetor[j+1];
59                  vetor[j+1] = aux;
60                  troca = 1;
61              }
62          }
63      }
64  }

```

```

67  /* *** Codifique aqui a funcao selectSort *** */
68  void selectSort (int *vetor, int N) {
69      int menor, aux, i, j;
70      for (i = 0; i < N-1; i++) {
71          menor = i;
72
73          for (j = i+1; j < N; j++)
74              if (vetor[j] < vetor[menor])
75                  menor = j;
76
77          if (menor != i) {
78              aux = vetor[i];
79              vetor[i] = vetor[menor];
80              vetor[menor] = aux;
81          }
82      }
83  }

```

```

86  /* *** Codifique aqui a funcao insertSort *** */
87  void insertSort (int *vetor, int N) {
88      int aux, i, j;
89      for (i = 1; i < N; i++) {
90          aux = vetor[i];
91
92          for (j = i-1; (j >= 0) && (aux < vetor[j]); j--)
93              vetor[j+1] = vetor[j];
94
95          vetor[j+1] = aux;
96      }
97  }

```

```
103 int main (void) {
104
105     int *base;
106     int *ordenado;
107
108     int n, i;
109
110     /* gerando dados para teste */
111     n = 10;
112     base = (int*) malloc (n * sizeof(int));
113     ordenado = (int*) malloc (n * sizeof(int));
114
115     for (i = 0; i < n; i++)
116         base[i] = rand() % 80;
117
118     /* testando Bubble sort */
119     printf("\nbubble sort version 1:\n");
120     copyVector(base, ordenado, n);
121     printVector(ordenado, n);
122     bubbleSort(ordenado, n);
123     printVector(ordenado, n);
124
125     printf("\nbubble sort version 2:\n");
126     copyVector(base, ordenado, n);
127     printVector(ordenado, n);
128     bubbleSort2(ordenado, n);
129     printVector(ordenado, n);
130
131     /* testando Select sort */
132     printf("\nselect sort:\n");
133     copyVector(base, ordenado, n);
134     printVector(ordenado, n);
135     selectSort(ordenado, n);
136     printVector(ordenado, n);
137
138     /* testando Insert sort */
139     printf("\ninsert sort:\n");
140     copyVector(base, ordenado, n);
141     printVector(ordenado, n);
142     insertSort(ordenado, n);
143     printVector(ordenado, n);
144
145     printf("\n");
146     free(base);
147     free(ordenado);
148     return 0;
149 }
```