NAME:	Kunal Bhatia
UID:	2021300010
SUBJECT:	DAA
EXPERIMENT NO:	02
DATE OF PERFORMANCE:	02/2/23
DATE OF SUBMISSION:	09/2/23
AIM:	Experiment on finding the running time of an algorithm.
Theory:	**Details**: The understanding of running time of algorithms is explored by implementing two basic sorting algorithms namely Insertion and Selection sorts. These algorithms work as follows. **Insertion sort:** It works similar to the sorting of playing cards in hands. It is assumed that the first card is already sorted in the card game, and then we select an unsorted card. If the selected unsorted card is greater than the first card, it will be placed at the right side; otherwise, it will be placed at the left side. Similarly, all unsorted cards are taken and put in their exact place. **Selection sort:** It first finds the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. In this algorithm, the array is divided into two parts, first is the sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and the unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right. In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted. **Problem Definition & Assumptions:** For this experiment, you need to implement two sorting algorithms namely Insertion and Selection sort methods. Compare these algorithms based on time and space complexity. Time required sorting algorithms can be performed using high_resolution_clock::now() under namespace std::chrono. You have to generate 1,00,000 integer numbers using C/C++ Rand function and save them in a text file. Both the sorting algorithm uses these 1,00,000 integer numbers as input as follows. Each sorting algorithm sorts a block of 100 integer numbers with array indexes numbers A[0..99], A[0..199], A[0..299],..., A[0..99999]. You need to use high_resolution_clock::now() function to find the time required for 100, 200, 300....

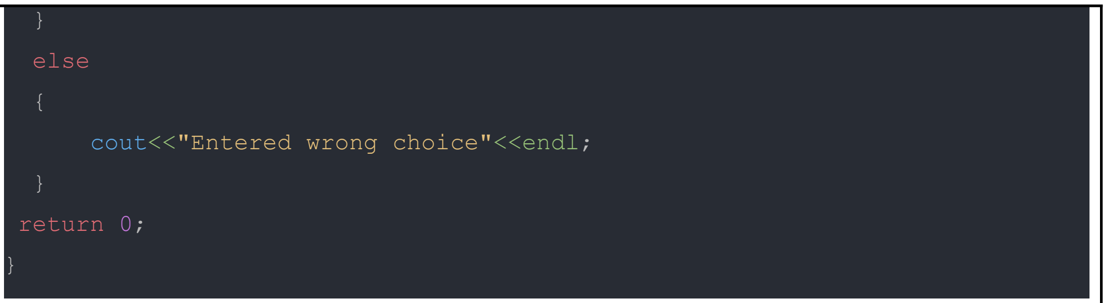
	100000 integer numbers. Finally, compare two algorithms namely Insertion and Selection by plotting the time required to sort 100000 integers using LibreOffice Calc/MS Excel. The x-axis of the 2-D plot represents the block no. of 1000 blocks. The y-axis of the 2-D plot represents the running time to sort 1000 blocks of 100,200,300,...,100000 integer numbers. **<u>Note</u>**: You have to use C/C++ file processing functions for reading and writing randomly generated 100000 integer numbers.
Algorithm:	Insertion Sort Function: ● Iterate from arr[1] to arr[N] over the array. ● Compare the current element (key) to its predecessor. ● If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element. Selection Sort Function: ● Set the array's first element as the minimum. ● Compare the first and second elements. Assign the second element as a minimum if it is smaller than the first. ● Compare the third element to the minimum. If the third element is smaller, assign it as a minimum; otherwise, don't do any changes. The process continues until the final element of the array. ● Minimum is moved to the front of the unsorted list after each iteration. ● Indexing begins with the first unsorted element in each iteration. Steps are repeated until all the elements are placed at their correct positions. GetInput Function: ● Function to make 100000 random numbers to sort and put into a text file Readfile Function: ● Function to read numbers from the text file Main Function: 1.Make a menu driven function and ask user his choice of sorting technique

	2.If insertion sort, call the function and calculate the time interval at every 100 numbers getting sorted up to 1000 times/block. 3.If Selection sort, call the function and calculate the time interval at every 100 numbers getting sorted up to 1000 times/block. 4.Else if, invalid input.

Code:

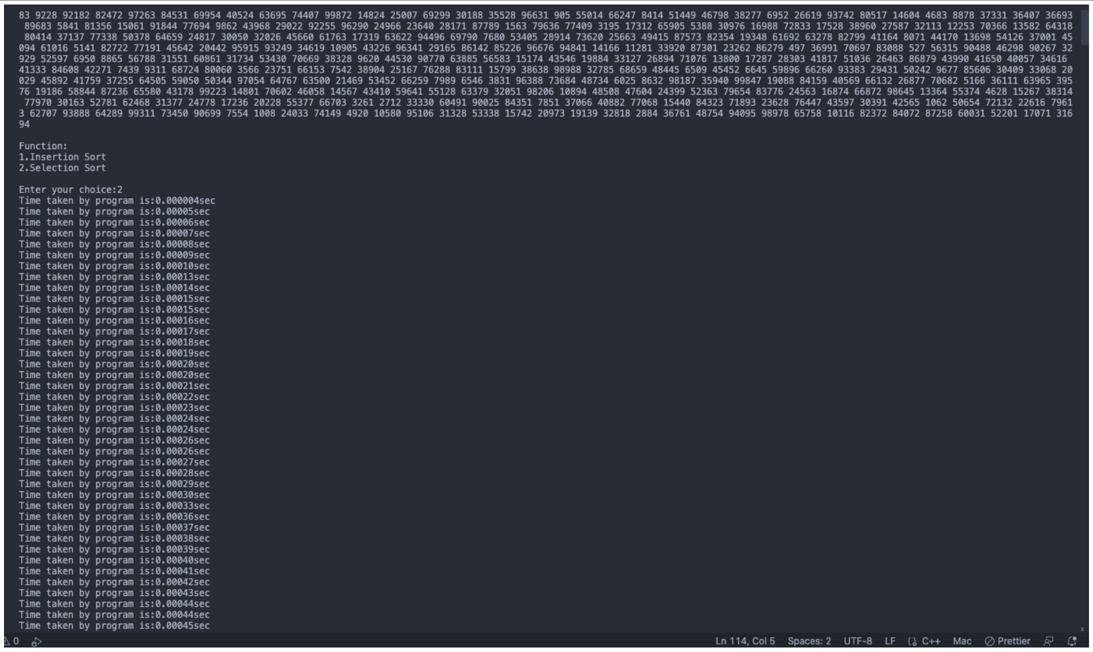
```
#include<iostream>
#include<fstream>
#include<time.h>
#include<cstdlib>
#include<fstream>
#include<string.h>
#include<chrono>
#include<numeric>
#include<iomanip>
using namespace std;

void getInput()
{
 ofstream fp;
 fp.open("input.text");
 for(int i=0;i<100000;i++)
 fp<< rand()%100000 << endl;
 fp.close();
}

void readfile(int arr[])
{
 int i=0;
 ifstream fp;
 fp.open("input.text");
 while(fp.good())
 {
   fp >> arr[i];
   i++;
 }
 fp.close();
```

```cpp
}
void insertionsort(int arr[],int i,int stop)
{
   int temp=i;
   if(i==stop)
   return;
   int curr=arr[i];
   temp--;
   while(temp>=0 && arr[temp]>curr)
   {
     arr[temp+1]=arr[temp];
     temp--;
   }
   arr[temp+1]=curr;
   insertionsort(arr,i+1,stop);
}


void selectionsort(int *arr,int size){
for(int i=0;i<size;i++){
   int min_idx = i;
   for(int j=min_idx+1;j<size;j++)
       if(arr[min_idx] > arr[j])
           min_idx = j;
   if(min_idx != i){
       int temp = arr[i];
       arr[i] = arr[min_idx];
       arr[min_idx] = temp;
   }
}
}

void printArray(int arr[], int n)
{
   int i;
   for (i = 0; i < n; i++)
       cout << arr[i] << " ";
   cout << endl;
}




int main()
{
 int i;
 int stop,limit;
int arr[100000];
clock_t start,end;
double time_taken=0;
getInput();
readfile(arr);
printf("The data:\n");
printArray(arr,100000);
printf("\nFunction:\n1.Insertion Sort\n2.Selection Sort\n");
printf("\nEnter your choice:");
 int ch;
 scanf("%d", &ch);
 if(ch==1)
 {
     time_taken=0;
     i=1;
     stop=100;
     while(i<=1000)
     {
     start=clock();
     insertionsort(arr,0,stop);
     end=clock();
     time_taken=time_taken+(double(end-start)/double(CLOCKS_PER_SEC));
     cout<<"Time taken by program
is:"<<fixed<<time_taken<<setprecision(5);
     cout<<"sec"<<endl;
     stop=stop+100;
     i++;
     }
     return 0;
 }
 else if(ch==2)
 {
     i=1;
     limit=100;
     while(i<=1000)
     {
     start=clock();
     selectionsort(arr,limit);
     end=clock();
     time_taken=(double(end-start)/double(CLOCKS_PER_SEC));
     cout<<fixed<<time_taken<<setprecision(5);
     cout<<""<<endl;
     limit=limit+100;
     i++;
     }
     return 0;
```

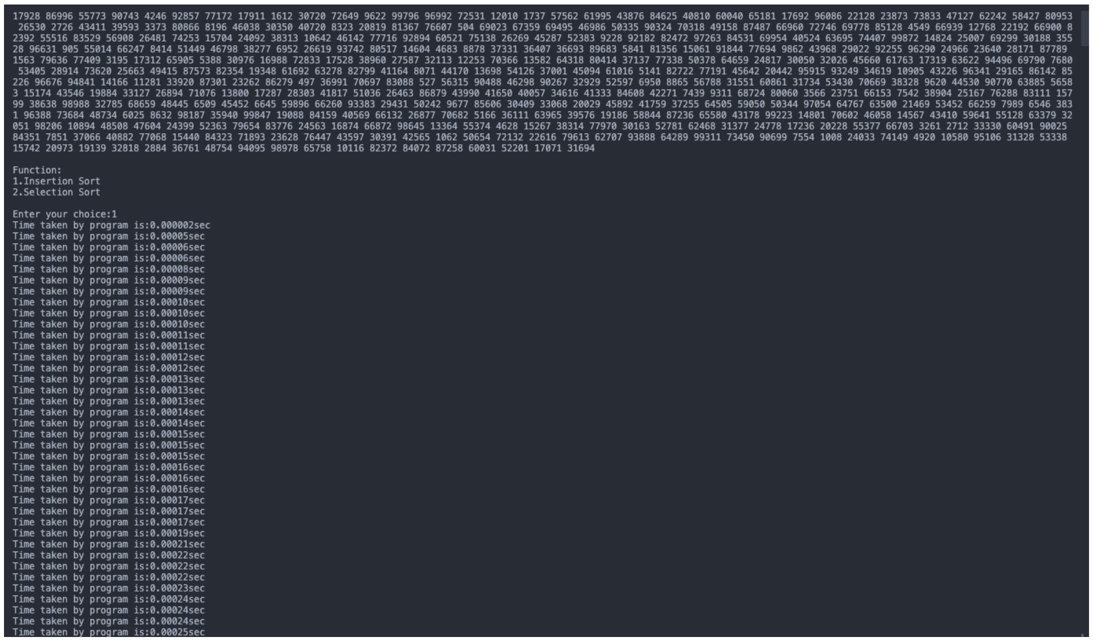


```cpp
  }
  else
  {
       cout<<"Entered wrong choice"<<endl;
  }
 return 0;
}
```

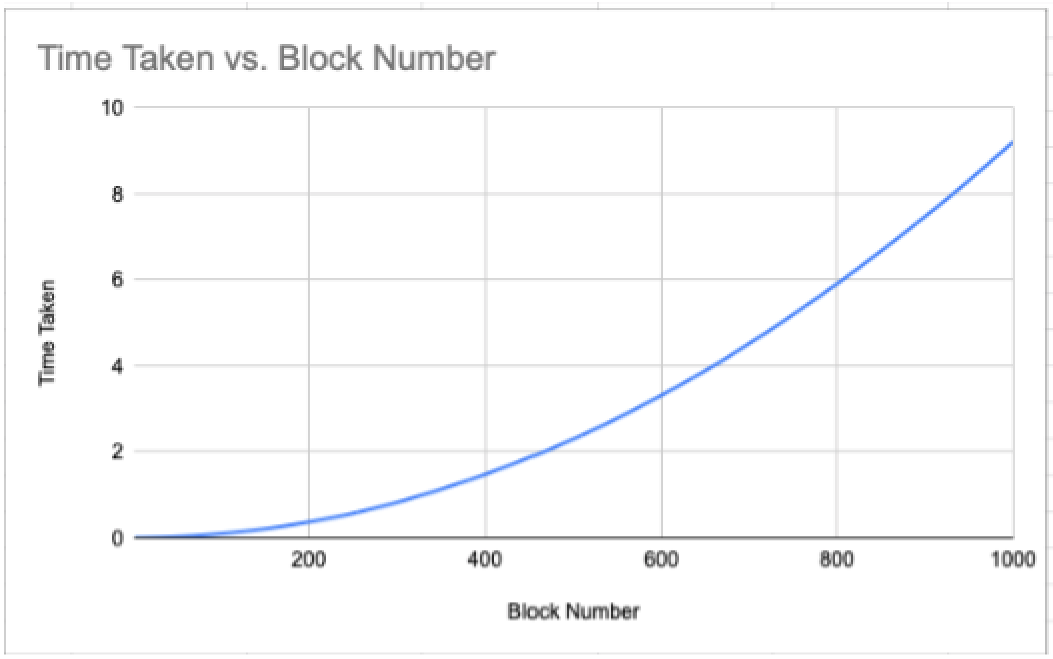
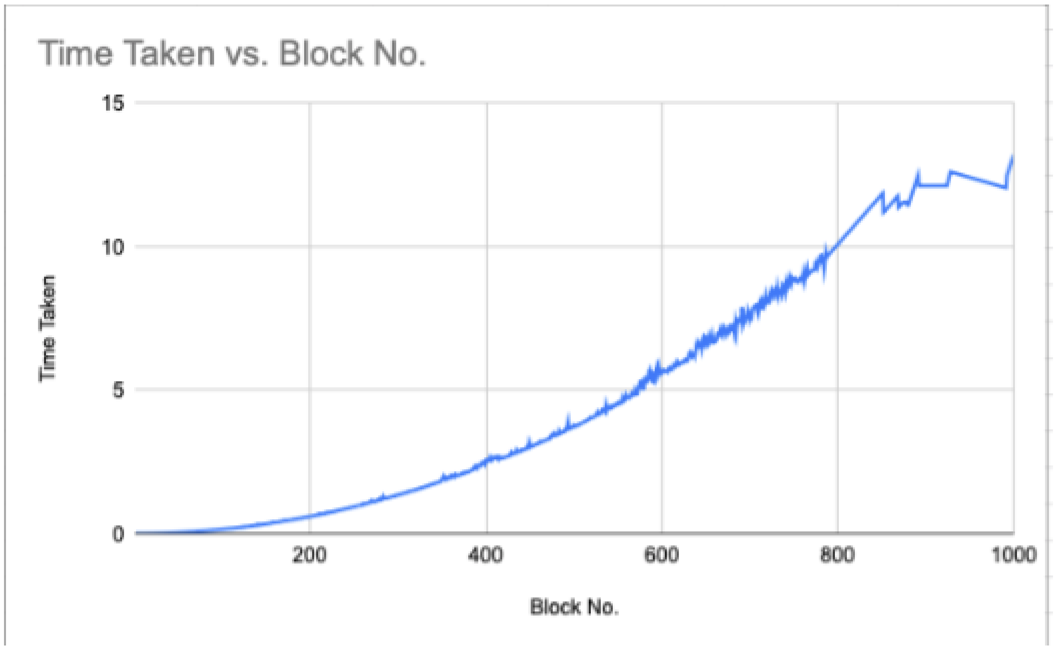
Results:



```
83 9228 92182 82472 97263 84531 69954 40524 63695 74407 99872 14824 25007 69299 30188 35528 96631 905 55014 66247 8414 51449 46798 38277 6952 26619 93742 80517 14604 4683 8878 37331 36407 36693
 89683 5841 81356 15061 91844 77694 9862 43968 29022 92255 96290 24966 23640 28171 87789 1563 79636 77409 3195 17312 65905 5388 30976 16988 72833 17528 38960 27587 32113 12253 70366 13582 64318
 80414 37137 77338 50378 64659 24817 30050 32026 45660 61763 17319 63622 94496 69790 7680 53405 28914 73620 25663 49415 87573 82354 19348 61692 63278 82799 41164 8071 44170 13698 54126 37001 45
094 61016 5141 82722 77191 45642 20442 95915 93249 34619 10905 43226 96341 29165 86142 85226 96676 94841 14166 11281 33920 87301 23262 86279 497 36991 70697 83088 527 56315 90488 46298 90267 32
929 52597 6950 8865 56788 31551 60861 31734 53430 70669 38328 9620 44530 90770 63885 56583 15174 43546 19884 33127 26894 71076 13800 17287 28303 41817 51036 26463 86879 43990 41650 40057 34616
41333 84608 42271 7439 9311 68724 80060 3566 23751 66153 7542 38904 25167 76288 83111 15799 38638 98988 32785 68659 48445 6509 45452 6645 59896 66260 93383 29431 50242 9677 85606 30409 33068 20
029 45892 41759 37255 64505 59050 50344 97054 64767 63500 21469 53452 66259 7989 6546 3831 96388 73684 48734 6025 8632 98187 35940 99847 19088 84159 40569 66132 26877 70682 5166 36111 63965 395
76 19186 58844 87236 65580 43178 99223 14801 70602 46058 14567 43410 59641 55128 63379 32051 98206 10894 48508 47604 24399 52363 79654 83776 24563 16874 66872 98645 13364 55374 4628 15267 38314
 77970 30163 52781 62468 31377 24778 17236 20228 55377 66703 3261 2712 33330 60491 90025 84351 7851 37066 40882 77068 15440 84323 71893 23628 76447 43597 30391 42565 1062 50654 72132 22616 7961
3 62707 93888 64289 99311 73450 90699 7554 1008 24033 74149 4920 10580 95106 31328 53338 15742 20973 19139 32818 2884 36761 48754 94095 98978 65758 10116 82372 84072 87258 60031 52201 17071 316
94

Function:
1.Insertion Sort
2.Selection Sort

Enter your choice:2
Time taken by program is:0.000004sec
Time taken by program is:0.00005sec
Time taken by program is:0.00006sec
Time taken by program is:0.00007sec
Time taken by program is:0.00008sec
Time taken by program is:0.00009sec
Time taken by program is:0.00010sec
Time taken by program is:0.00013sec
Time taken by program is:0.00014sec
Time taken by program is:0.00015sec
Time taken by program is:0.00015sec
Time taken by program is:0.00016sec
Time taken by program is:0.00017sec
Time taken by program is:0.00018sec
Time taken by program is:0.00019sec
Time taken by program is:0.00020sec
Time taken by program is:0.00020sec
Time taken by program is:0.00021sec
Time taken by program is:0.00022sec
Time taken by program is:0.00023sec
Time taken by program is:0.00024sec
Time taken by program is:0.00024sec
Time taken by program is:0.00026sec
Time taken by program is:0.00026sec
Time taken by program is:0.00027sec
Time taken by program is:0.00028sec
Time taken by program is:0.00029sec
Time taken by program is:0.00030sec
Time taken by program is:0.00033sec
Time taken by program is:0.00036sec
Time taken by program is:0.00037sec
Time taken by program is:0.00038sec
Time taken by program is:0.00039sec
Time taken by program is:0.00040sec
Time taken by program is:0.00041sec
Time taken by program is:0.00042sec
Time taken by program is:0.00043sec
Time taken by program is:0.00044sec
Time taken by program is:0.00044sec
Time taken by program is:0.00045sec
```



```
17928 86996 55773 90743 4246 92857 77172 17911 1612 30720 72649 9622 99796 96992 72531 12010 1737 57562 61995 43876 84625 40810 60040 65181 17692 96086 22128 23873 73833 47127 62242 58427 80953
 26530 2726 43411 39593 3373 80866 8196 46038 30350 40720 8323 20819 81367 76607 504 69023 67359 69495 46986 50335 90324 70318 49158 87487 66960 72746 69778 85128 4549 66939 12768 22192 66900 8
2392 55516 83529 56908 26481 74253 15704 24092 38313 10642 46142 77716 92894 60521 75138 26269 45287 52383 9228 92182 82472 97263 84531 69954 40524 63695 74407 99872 14824 25007 69299 30188 355
28 96631 905 55014 66247 8414 51449 46798 38277 6952 26619 93742 80517 14604 4683 8878 37331 36407 36693 89683 5841 81356 15061 91844 77694 9862 43968 29022 92255 96290 24966 23640 28171 87789
1563 79636 77409 3195 17312 65905 5388 30976 16988 72833 17528 38960 27587 32113 12253 70366 13582 64318 80414 37137 77338 50378 64659 24817 30050 32026 45660 61763 17319 63622 94496 69790 7680
 53405 28914 73620 25663 49415 87573 82354 19348 61692 63278 82799 41164 8071 44170 13698 54126 37001 45094 61016 5141 82722 77191 45642 20442 95915 93249 34619 10905 43226 96341 29165 86142 85
226 96676 94841 14166 11281 33920 87301 23262 86279 497 36991 70697 83088 527 56315 90488 46298 90267 32929 52597 6950 8865 56788 31551 60861 31734 53430 70669 38328 9620 44530 90770 63885 5658
3 15174 43546 19884 33127 26894 71076 13800 17287 28303 41817 51036 26463 86879 43990 41650 40057 34616 41333 84608 42271 7439 9311 68724 80060 3566 23751 66153 7542 38904 25167 76288 83111 157
99 38638 98988 32785 68659 48445 6509 45452 6645 59896 66260 93383 29431 50242 9677 85606 30409 33068 20029 45892 41759 37255 64505 59050 50344 97054 64767 63500 21469 53452 66259 7989 6546 383
1 96388 73684 48734 6025 8632 98187 35940 99847 19088 84159 40569 66132 26877 70682 5166 36111 63965 39576 19186 58844 87236 65580 43178 99223 14801 70602 46058 14567 43410 59641 55128 63379 32
051 98206 10894 48508 47604 24399 52363 79654 83776 24563 16874 66872 98645 13364 55374 4628 15267 38314 77970 30163 52781 62468 31377 24778 17236 20228 55377 66703 3261 2712 33330 60491 90025
84351 7851 37066 40882 77068 15440 84323 71893 23628 76447 43597 30391 42565 1062 50654 72132 22616 79613 62707 93888 64289 99311 73450 90699 7554 1008 24033 74149 4920 10580 95106 31328 53338
15742 20973 19139 32818 2884 36761 48754 94095 98978 65758 10116 82372 84072 87258 60031 52201 17071 31694

Function:
1.Insertion Sort
2.Selection Sort

Enter your choice:1
Time taken by program is:0.000002sec
Time taken by program is:0.00005sec
Time taken by program is:0.00006sec
Time taken by program is:0.00006sec
Time taken by program is:0.00008sec
Time taken by program is:0.00009sec
Time taken by program is:0.00009sec
Time taken by program is:0.00010sec
Time taken by program is:0.00010sec
Time taken by program is:0.00010sec
Time taken by program is:0.00011sec
Time taken by program is:0.00011sec
Time taken by program is:0.00012sec
Time taken by program is:0.00012sec
Time taken by program is:0.00013sec
Time taken by program is:0.00013sec
Time taken by program is:0.00013sec
Time taken by program is:0.00014sec
Time taken by program is:0.00014sec
Time taken by program is:0.00015sec
Time taken by program is:0.00015sec
Time taken by program is:0.00015sec
Time taken by program is:0.00016sec
Time taken by program is:0.00016sec
Time taken by program is:0.00016sec
Time taken by program is:0.00017sec
Time taken by program is:0.00017sec
Time taken by program is:0.00017sec
Time taken by program is:0.00019sec
Time taken by program is:0.00021sec
Time taken by program is:0.00022sec
Time taken by program is:0.00022sec
Time taken by program is:0.00022sec
Time taken by program is:0.00023sec
Time taken by program is:0.00024sec
Time taken by program is:0.00024sec
Time taken by program is:0.00024sec
Time taken by program is:0.00025sec
```

Graph:	 
Conclusion:	I understood the time complexity difference between the two sorting:insertion and selection.